REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-01-

0176

1. REPORT DATE *(DD-MM-YYYY)* 02-22-2001	2. REPORT TYPE Final Technical Report	3. DATES COVERED *(From - To)* 12/01/1997 - 11/30/2000

4. TITLE AND SUBTITLE Self-Designing Control Systems for Piloted and Uninhabited Aerial Vehicles	5a. CONTRACT NUMBER FQ8671-9800236
	5b. GRANT NUMBER F49620-98-1-0013
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Marc Bodson	5d. PROJECT NUMBER 2304/AS
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Department of Electrical Engineering University of Utah 50 S Central Campus Dr Rm 3280 Salt Lake City, UT 84112-9206	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Dr Marc Jacobs AFOSR/NM 801 North Randolph Street, Room 732 Arlington, VA 22203-1977	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFOSR)
NOTICE OF TRANSMITTAL DTIC. THIS TECHNICAL REPORT
HAS BEEN REVIEWED AND IS APPROVED FOR PUBLIC RELEASE
LAW AFR 190-12. DISTRIBUTION IS UNLIMITED.

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes.

13. SUPPLEMENTARY NOTES
The views and conclusions contained in the report are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

14. ABSTRACT
The main advantage of self-designing flight control systems is their ability to optimize performance automatically, resulting in a substantial reduction of the cost and time needed for control law development. Self-designing control systems also reconfigure automatically after failures and damages, yielding greater chances of survival in dangerous conditions. A self-designing nonlinear autopilot was developed to interface with the high-level path planning of a UAV. The control algorithm is distinct from conventional autopilots in that it is not based on a known, linearized model of the aircraft. Instead, the algorithm compensates for nonlinear dynamic effects and adjusts its parameters automatically, exploiting the reconfiguration capabilities of an inner control loop designed using adaptive methods. A new control allocation algorithm was also developed, based on the direct allocation method of Durham. A special representation using spherical coordinates was used to speed-up the computations that must be performed at a high sampling rate. The direct allocation method was also extended to a class of systems that had previously been excluded, namely those for which some independent control surfaces produce linearly dependent moments. Finally, fast algorithms for optimal control allocation were developed based on linear programming techniques. It was observed that significant improvements in performance could be obtained at the cost of only modest increases in computations.

15. SUBJECT TERMS
Flight control systems, reconfigurable control, adaptive control, uninhabited aerial vehicles, UAV, control allocation, multivariable systems, constrained dynamic systems.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Professor Marc Bodson
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	70	19b. TELEPHONE NUMBER *(include area code)* (801) 581 8590

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

FINAL REPORT

SELF-DESIGNING CONTROL SYSTEMS FOR PILOTED AND UNINHABITED AERIAL VEHICLES

AFOSR F49620-98-1-0013

Marc Bodson
University of Utah, Electrical Engineering
50 S Central Campus Dr Rm 3280
Salt Lake City, UT 84112-9206

Objectives

The objective of the project was to advance the state-of-the-art in self-designing flight control systems. Applications to both piloted and remotely piloted vehicles were considered. The main advantage of self-designing control systems over conventional systems is their ability to optimize performance automatically, resulting in a substantial reduction of the cost and time needed for control law development. Self-designing control systems also reconfigure automatically after failures and damages, yielding greater chances of survival in dangerous conditions.

Applications to Air Force Systems include uninhabited aerial vehicles (UAV's), tailless fighter aircraft, and military transport aircraft. Self-designing control systems are particularly attractive for combat UAV's, because these aircraft operate in the most hostile environments and must be low-cost. A particular problem with future aircraft is also that they will be equipped with a large number of control effectors to provide redundancy in the case of failures or damages. An optimal distribution of the control requirements may be required to perform challenging maneuvers or to recover from unusual flight conditions.

Accomplishments

Parameter Identification

The problem of parameter identification for self-designing flight control is similar to the off-line identification problem, but involves additional difficulties which may be categorized as follows:

- *Lack of control over the actuator signals.* Identification performance may be poor because the signals are determined by the control law and may exhibit undesirable characteristics, including: (1) high levels of correlation between the control signals and the aircraft states ; (2) long periods of quiescence; (3) highly coupled longitudinal and lateral motions; (4) excitation of nonlinear dynamics.

- *Increased number of parameters for failed aircrafts.* Aircrafts often loose their symmetry after failures, requiring that more parameters be identified.

- *Real-time operation.* The computational requirements of a parameter identification scheme for reconfigurable control must be compatible with available computers, and the procedure must be performed with minimal supervision.

Based on a concept proposed by Ward & Barron (D. Ward & R. Barron, *Self Designing Flight Control Using Modified Sequential Least-Squares Parameter Estimation and Optimal Receding Horizon Control Laws*, Technical Report, Barron Assoc., Charlottesville, VA, March 1994), a recursive algorithm was developed that was found to perform well in reconfigurable control problems. Analytical results provided estimates of convergence rates as a function of the information available in the data. The algorithm was used with success in simulations of a twin-engine fighter aircraft, and flight tests were performed using the Vista F-16 aircraft, under the leadership of Barron Associates, Inc. (the PI acted as an advisor in that project). The tests demonstrated the practical feasibility of on-line reconfiguration algorithms, with the results of the experiments reported in [1]. The adaptive algorithm was also tested with success in the laboratory of the principal investigator on a vibration control problem [2]. In addition, the adaptive algorithm was adapted for NASA's IDACS program, and Fortran code was sent to *Boeing* for that project.

Self-Designing Control Algorithms

Advanced simulation capabilities were developed to evaluate the performance of the control algorithms proposed in the course of the project. A detailed F-16 simulation code was written, with the following features: (a) 6 degree-of-freedom aircraft model with nonlinear aerodynamic tables, (b) first-order actuator models with position/rate saturation and locked/floating failures, (c) atmospheric model and engine thrust model, (d) turbulence model (steady wind and Dryden spectrum), and (e) sensor noise model. The simulation was adapted and extended from the code available in B. Stevens & F. Lewis, *Aircraft Control and Simulation*, Wiley-Interscience, New York, NY, 1992.

Models of a C-17 transport aircraft and of a tailless aircraft were also incorporated in the code. These models were simpler in that the aerodynamics were linear (they matched a state-space model at a given flight condition) and thrust effects were not modeled. However, nonlinearities due to rotational motions, dynamic pressure, and actuator saturation, were included. The C-17 model was obtained in the course of a contract with *McDonnell Douglas Co.* (now *Boeing Co.*) in which software for real-time identification of parameters of C-17 and B-777 aircrafts was delivered. The tailless aircraft model represented *Lockheed Martin*'s *ICE* (innovative control effector) aircraft, obtained from J. Buffington, "Tailless Aircraft Control Allocation," *Proc. of the AIAA Guidance, Navigation, and Control Conference*, New Orleans, LA, pp. 737-747, 1997.

A 3D-visualization capability was added to the software, so that the results of simulations could be viewed on a computer monitor and provide additional insight into the characteristics of the control laws. A real-time operating mode was also incorporated, with joystick inputs, making it possible to investigate a wide range of maneuvers in an environment similar to a flight simulator. Fig. 1 is a screenshot showing a frame of the simulation of an F-16 with a missing left horizontal tail surface. An advanced model reference adaptive control system developed earlier with AFOSR support was incorporated in the code, after adjustments (see M. Bodson & J. Groszkiewicz, "Multivariable Adaptive Algorithms for Reconfigurable Flight Control," *IEEE Trans. on Control Systems Technology,* vol. 5, no. 2, pp. 217-229, 1997). The command limiting technique described in [3] was also incorporated. Although these algorithms had been tested before in off-line simulations, their evaluation in real-time "piloted" simulations further demonstrated their excellent performance and robustness properties.

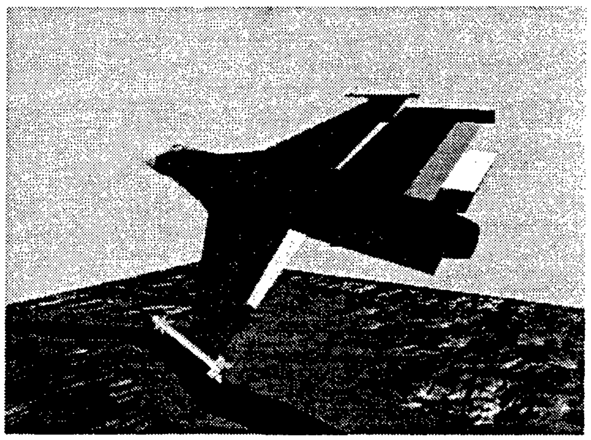

Fig. 1: Screenshot of an F-16 simulation
with missing left horizontal tail

The original reconfigurable control laws were developed for the tracking of pitch rate, roll rate, and yaw rate commands. Although the use of angular rate commands is appropriate for piloted air vehicles, higher level of commands are necessary for UAV's. A path planning algorithm may assume that the velocity vector is specified, or that the commanded variables are those typical of an autopilot, *e.g.,* altitude, heading, and sideslip. A self-designing, nonlinear autopilot was developed, based on a pitch/roll/yaw reconfigurable control law. Its structure is shown in Fig. 2. Some of the challenges encountered in the development of the control algorithm were the nonlinear effects associated with the bank angle, and the need to develop a control strategy that was self-designing. The nonlinear autopilot estimated the required parameters, and compensated for the nonlinear couplings inherent in the aircraft dynamics.

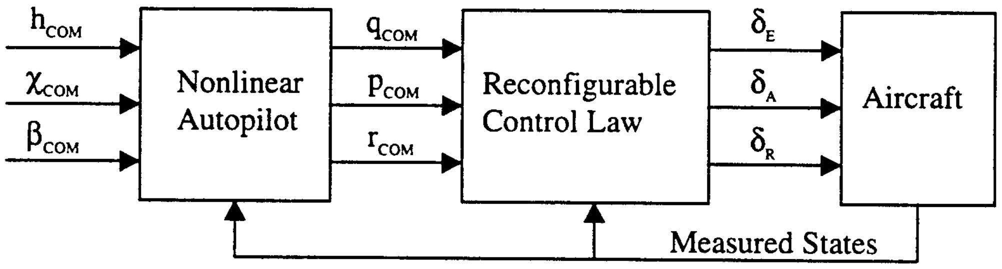

Figure 2: Nonlinear autopilot using a reconfigurable inner loop

Fig. 3 shows the results of a simulation of the nonlinear autopilot combined with a reconfigurable control law, as applied to the high-fidelity F-16 simulation model. In the simulation, the aircraft was commanded to fly at an altitude of 1000 ft and a speed of 502 ft/s. Changes of heading between 0 and 45 degrees were commanded at regular intervals. The sideslip command was zero. The top/left plot of Fig. 3 shows the heading command (dashed) and the heading angle response (solid). In the transient portion of the heading response, the bank angle reaches a (limited) value of 45 degrees, resulting in a linearly increasing heading angle. The top/right response shows the response in altitude. The variations of altitude are small, considering the steep bank angle associated with the maneuver.

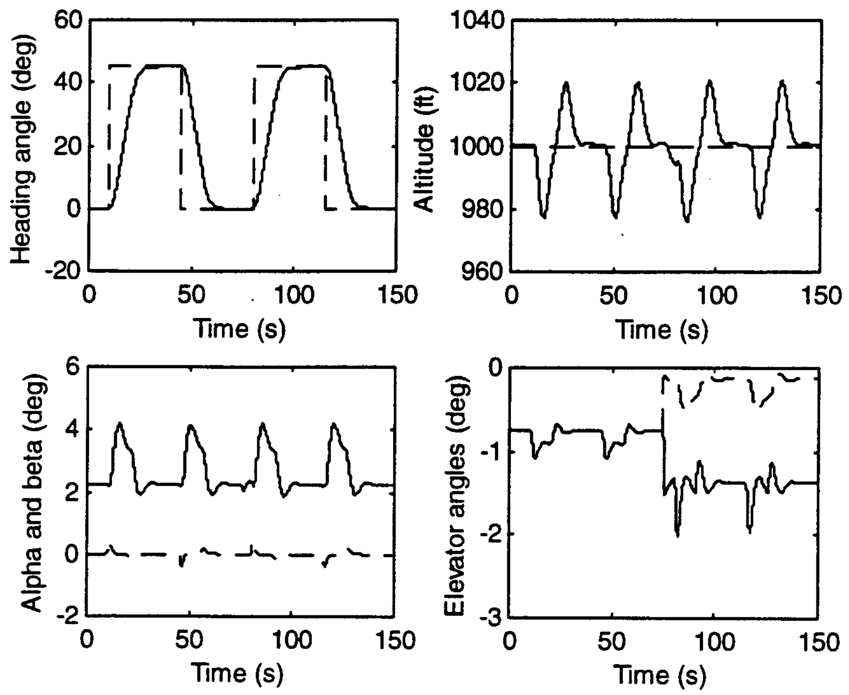

Figure 3: Simulation results of the self-designing nonlinear autopilot

The bottom/left plot shows the angle of attack (solid) and the angle of sideslip (dashed). Note that the control law forces significant increases in angle of attack, in order to compensate for the loss of lift and sustain the altitude. The result is obtained through the estimation of the aircraft's lift parameters in real-time. At the same time, low angles of sideslip are maintained. The control law automatically performs turn coordination, which requires application of carefully balanced commands in all three axes.

The bottom/right plot shows the right elevator angle (solid) and the left elevator angle (dashed). Mid-way through the simulation, a floating elevator failure was simulated by moving the left tail surface to the local angle of attack. As can be seen from the last plot, the inner reconfigurable control loop successfully estimated the parameters of the system: the right elevator position shifted to restore trim, and larger deflections were applied to compensate for the loss of effectiveness of the left elevator. Overall, the effect of the failure is not observed on the flight path variables. The details of the algorithm are discussed in a report under preparation [10]. An M.S. thesis is also to be completed soon and will include an extensive evaluation of the autopilot [11].

Direct Allocation Using Spherical Coordinates

Control allocation is the problem of distributing control requirements among redundant control surfaces. The problem is particularly important for tailless aircrafts (because current designs involve a large number of control surfaces) and for reconfigurable control laws (because control surfaces may need to be commanded separately if the maximum capabilities of the aircraft are to be exploited). This part of the research was performed with the additional support of an AASERT award, "Constrained Control Allocation Methods for Reconfigurable Flight Control Laws" (Grant: AFOSR F49620-97-1-0405).

For model reference control laws, as well as for other reconfigurable control laws, the control allocation problem is to find a control input u is such that

$$(CB)u = a_d$$

where a_d is a desired vector and the matrix CB has more columns than rows (for example 3 rows and 11 columns). The difficulty in practice is that the vector u is constrained, so that

$$u_{\min,i} \le u_i \le u_{\max,i} \qquad \text{for } i = 1,\ldots,m$$

or, $u_{\min} \le u \le u_{\max}$, in vector form. Given these limits, an exact solution may not exist, despite the redundancy.

As part of the project, a new control allocation algorithm was developed, based on the *direct allocation* method of Durham. The direct allocation method was chosen because it utilized all of the attainable moment set. The objective of direct allocation is to find a control vector u that results in the best approximation of the vector a_d, *in the given direction.* The implicit assumption is that directionality is an important characteristic of multivariable control systems, and of flight control systems in particular.

At the core of Durham's method was the computation of the set of attainable moments (or set of attainable accelerations here). Fig. 4 shows the sets of attainable accelerations for the C-17 transport aircraft model (on the left) and for the tailless fighter aircraft model (on the right). The axes of the plots are the pitch, roll, and yaw accelerations in deg/s^2. The boundaries of the sets specify the maximum accelerations obtainable under the control limits, assuming linear models for the aircraft dynamics. What makes the control allocation problem difficult is the large number of actuators: 11 for the tailless aircraft and 16 for the C-17 aircraft. A computer code was developed that produces the sets shown on Fig. 4 in a fraction of a second.

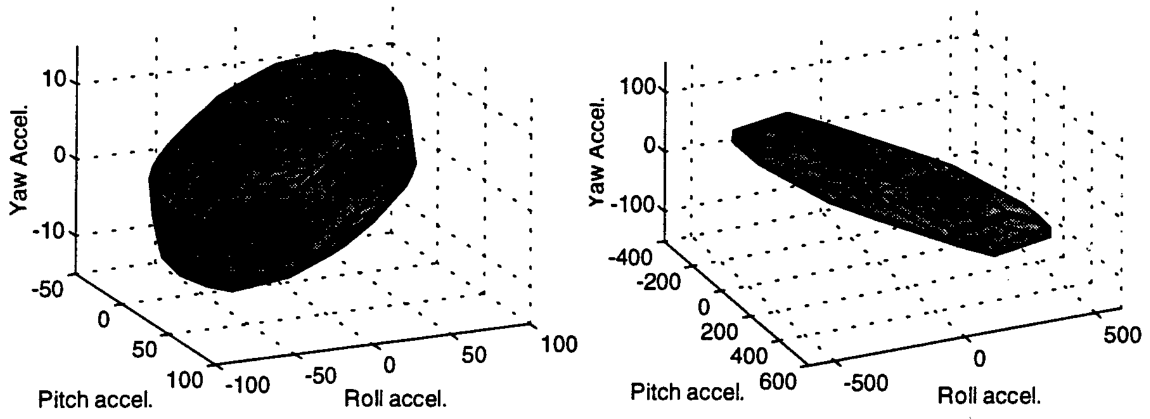

Fig. 4: Sets of attainable accelerations, C-17 aircraft (left) and tailless aircraft (right)

The set for the tailless aircraft is delimited by 78 facets, while the set for the C-17 is determined by 240 facets. The method developed by Durham and co-workers requires that, if one extracts any three columns of the *CB* matrix, the resulting matrix is nonsingular. This condition is satisfied for the C-17 model used for Fig. 4 (on the left) and implies that every facet is a parallelogram. For the tailless model, the condition is not satisfied. In fact, some columns of the matrix are even linearly dependent (because pitch thrust vectoring and pitch flaps only produce pitching accelerations). To obtain the plot shown on the right of Fig. 4, the original method was extended to relax the linear independence requirement [6]. Note that two polygonal facets are visible on the plot.

The most significant part of the computations in Durham's original method is performed to obtain the set of attainable accelerations. To compute the control signals after the set is obtained, one must determine the facet towards which the desired acceleration vector points, and then perform some simple computations. Since the determination of the set may be performed off-line, our work has been based on the observation that the method becomes a fast algorithm for control allocation guaranteeing the use of the maximum control authority, *if the applicable facet can be found rapidly*.

The use of spherical coordinates was investigated as a way to perform the search [5]. On Fig. 5 is a representation of the two sets of Fig. 4 in spherical coordinates. Note that, because the determination of the applicable facet is only dependent on the direction of the desired acceleration vector, the search problem is effectively a 2-dimensional problem.

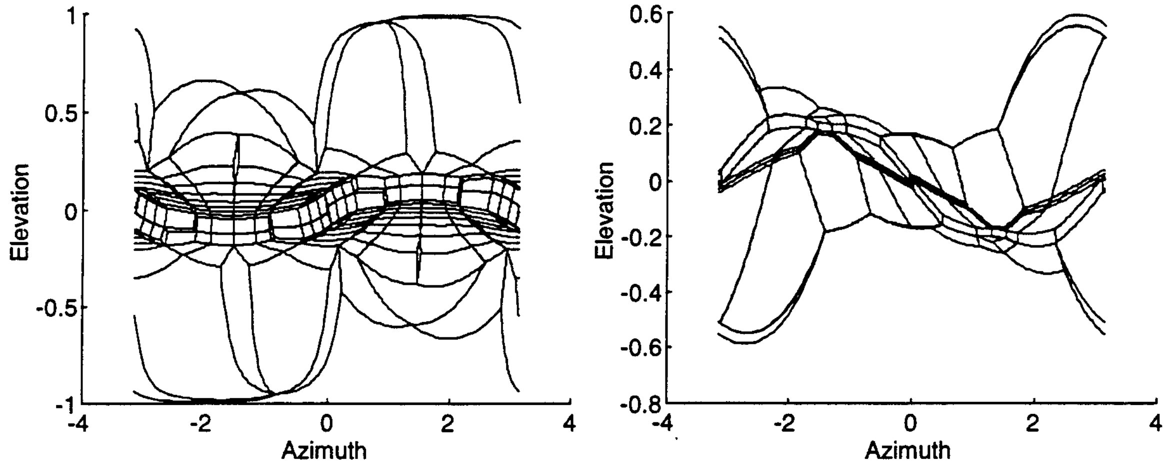

Fig. 5: Sets of attainable accelerations expressed in spherical coordinates, C-17 aircraft (left) and tailless aircraft (right)

Two methods were developed using the idea of spherical coordinates [5]. In the first method, ranges were pre-computed for the coordinates of the facets, defining boxes that contained the facets. The search was then similar to an exhaustive search of the facets, except that simple inequality tests were used to quickly eliminate facets from the search. For those facets that satisfied the box check, a more complicated test was performed. This test conclusively established whether the facet was the correct one. If the test was successful, the control input was rapidly obtained. The idea was that the more complicated test was only required for very few facets. In experiments with the C-17 model, the test was required once in 50% of the cases and less than three times in 95% of the cases. Overall, it was required an average of 1.97 times, as opposed to 49.7 times in the case of an exhaustive search.

A second method was also developed using the concept of spherical coordinates. In that method, a table of facets was created off-line, and the table was indexed by spherical coordinates. The determination of the applicable facet was then achieved by simple table look-up. This option required virtually no on-line computations and provided a guaranteed solution in a fixed time. Its drawback was a large memory requirement and longer off-line execution times.

The same techniques were applied to systems that did not satisfy the linear independence condition of the original method (systems with so-called *coplanar controls*). The tailless aircraft model was taken as an example [6]. As shown in Fig. 4, Durham's condition was relaxed for the determination of the attainable acceleration set by replacing the parallelogram facets by polygons. To determine the control vector u, two options were considered. The first used representations of the polygonal facets in spherical coordinates. This option turned out to be difficult to implement. A simpler option was explored that consisted in representing the polygonal facets as superimposed parallelogram facets. This option had the advantage of sharing many features with the original method. Given a desired acceleration, the table look-up method produced several sub-facets, each of them yielding a valid control input (the solution is not necessarily unique if there are coplanar controls). In order to ensure the continuity of the solution, the solution that was picked was the average of the solutions corresponding to each sub-facet.

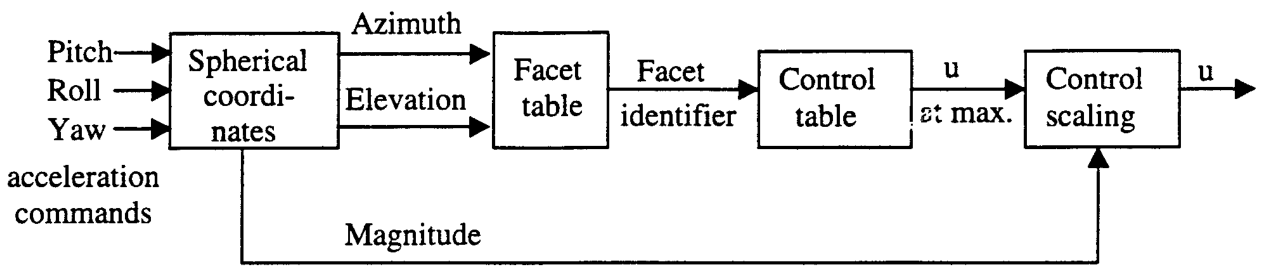

Fig. 6: Block diagram of a table look-up algorithm for direct allocation using spherical coordinates

Fig. 6 shows a block diagram representation of one of the possible implementations of Durham's method using the concepts proposed in [5] and [6]. The two inner blocks are look-up tables, while the outer blocks perform simple computations. The resulting algorithm has the advantage of being feasible in real-time and to guarantee the availability of all the control authority attainable. Its reliability is also excellent, as the computations are very simple and the results predictable. A drawback of the method is that it requires a substantial memory space. Some computations must also be performed off-line for the creation of the look-up tables and, in a reconfigurable control application, must be performed on-line. However, these "off-line" computations may be carried out at a lower computational rate, that is, at the rate associated with the variation of the adaptive parameters. A journal paper [7] has been prepared that summarizes the results of [5] and [6], and is provided in appendix.

Fast Algorithms for Optimal Control Allocation

Our work on the direct allocation method of Durham has led us to more general problem formulations, and to the solution of control allocation problems using optimization techniques. The objective of optimal formulations is to derive methods that make available all of the capabilities of the aircraft, and may be easily adapted to changing conditions (*e.g.*, changing actuator effectiveness due to damage or failure). Achieving all the control authority available may make the difference between a maneuver being realizable or not, and between an unusual condition being recoverable from or not.

Three formulations of optimal control allocation were posed:

1) Error Minimization Problem: given a matrix CB, find a vector u such that $J = \|CBu - a_d\|$ is minimized, subject to $u_{\min} \leq u \leq u_{\max}$.

2) Direct Allocation Problem: given a matrix CB, find a real number ρ and a vector u_1 such that J=ρ is maximized, subject to $(CB)u_1 = \rho a_d$ and $u_{\min} \leq u \leq u_{\max}$. If ρ>1, let $u = u_1/\rho$. Otherwise, let $u = u_1$.

3) Control Minimization Problem: given a matrix CB and a vector u_1 such that $u_{\min} \leq u_1 \leq u_{\max}$, find a vector u such that $J = \|u\|$ is minimized, subject to $(CB)u = (CB)u_1$ and $u_{\min} \leq u \leq u_{\max}$.

The first optimization problem is an intuitive formulation of control allocation. The second optimization problem is the direct allocation problem discussed earlier. The third problem is a secondary optimization objective to be satisfied if the solution of the primary objective (either the first or the second problem) is not unique. In that case, the solution of minimum norm is picked among all possible ones. The minimization of the norm of the vector u may be replaced by the minimization of the error between u and some preferred position.

The three constrained optimization problems are simple, from a modern computational perspective. However, computations must be performed at a high rate (around 100 Hz) and in the midst of many other control computations. Further, predictability and reliability are required for safety-critical systems, and human intervention is not possible. For this reason, approximations to the optimization problems have generally been implemented. Nevertheless, we have tested algorithms for the exact solution of all three control allocation problems, and found that the time required was of the same order of magnitude as that of the simpler methods [8]. Results have also shown significant performance improvements. The paper [8] is attached in Appendix. A significant research effort has also been aimed at applying interior-point methods for the solution of control allocation problems. The results of this effort will be reported in [12].

Flight Testing with Remote-Controlled Aircraft

A flight testing platform was built using commercial remote-controlled aircraft hardware. Two undergraduate students completed senior projects on this topic. Fig. 7 shows the twin-engine aircraft that was built. The nose cone developed by the students is visible on the photo. It provides air pressure, angle of attack and angle of sideslip measurements. Sensors also include gyroscopes and accelerometers. As opposed to other similar projects, our airframe is based on a commercial almost-ready-to-fly (ARF) R/C aircraft and low-cost instrumentation (no GPS or sophisticated on-board computer), with the objective of performing experiments that would not be considered feasible with conventional flight testing platforms. One of the undergraduate students working on this project continued on an M.S. program, and the results of the real-time identification and control reconfiguration experiments will be reported in [13].

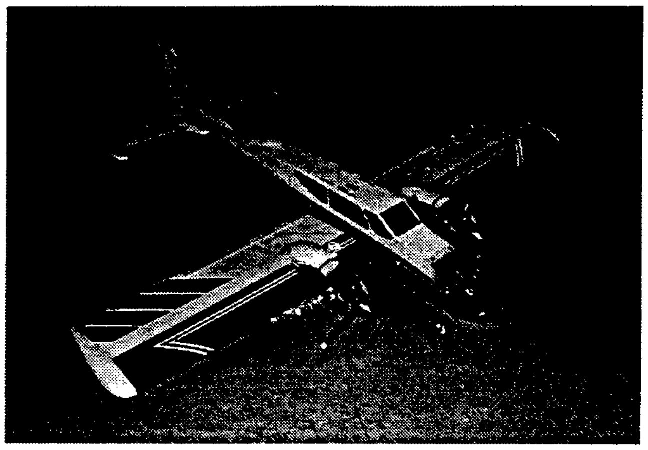

Fig. 7: Remote-controlled aircraft developed for flight control experiments

Personnel Supported

Faculty: Marc Bodson. Graduate Students: Mark Leatherwood, John Petersen (both partly supported through an AASERT award), Haiping Wu, Biqing Wu, and Eugene You. Dan Stevens and David Shore completed a B.S. project related to this project without salary support.

Publications During the Period of the Grant

References [1] to [6] are available from the public domain. References [7] and [8] are not available in the literature yet, and are attached in Appendix. Other references will become available as indicated. Information about all publications may be obtained by contacting the principal investigator, Professor Marc Bodson, by phone at (801) 581 8590, or e-mail using bodson@ee.utah.edu.

[1] D. Ward, J. Monaco, & M. Bodson, "Development and Flight Testing of a Parameter Identification Algorithm for Reconfigurable Flight Control, " *AIAA Journal of Guidance, Control, and Dynamics*, vol. 2, no. 6, pp. 948-956, 1998.
[2] M. Bodson, "An Adaptive Algorithm for the Tuning of Two Input Shaping Methods," *Automatica*, vol. 34, no. 6, pp. 771-776, 1998.
[3] M. Bodson & W. Pohlchuck, "Command Limiting in Reconfigurable Flight Control," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 4, pp. 639-646, 1998.
[4] J. Stephan, M. Bodson, & J. Lehoczky, "Calculation of Recoverable Sets for Systems with Input and State Constraints, " *Optimal Control Applications & Methods*, vol. 19, pp. 247-269, 1998.
[5] J. Petersen & M. Bodson, "Fast Control Allocation Using Spherical Coordinates," *Proc. of the AIAA Guidance, Navigation, and Control Conference*, Portland, OR, pp. 1321-1330, 1999.

[6] J. Petersen & M. Bodson, "Control Allocation for Systems with Coplanar Controls," *Proc. of the AIAA Guidance, Navigation, and Control Conference*, Denver, CO, paper AIAA 2000-4540, August 2000.
[7] J. Petersen & M. Bodson, "Fast Implementation of Direct Allocation with Extension to Coplanar Controls," submitted for publication to the *AIAA Journal of Guidance, Control, and Dynamics*, 2000.
[8] M. Bodson, "Evaluation of Optimization Methods for Control Allocation," submitted for presentation at the *AIAA Guidance, Navigation and Control Conference*, Montreal, Canada, August 2001.
[9] H. Wu, "Design of Active Vibration Control Algorithms using Linear Time-Invariant Equivalence," M.S. thesis, Department of Electrical Engineering, University of Utah, December 2000.
[10] M. Bodson, "A Self-Designing Nonlinear Autopilot," report to be completed May 2001.
[11] M. Leatherwood, "Self-Designing Control Systems for Aerial Vehicles," M.S. thesis, Department of Electrical Engineering, University of Utah, to be completed May 2001.
[12] J. Petersen, "Constrained Control Allocation for Reconfigurable Flight Control," Ph.D. thesis, Department of Electrical Engineering, University of Utah, to be completed August 2001.
[13] D. Shore, "Real-Time Identification and Control Reconfiguration of a Remotely Piloted Vehicle," M.S. thesis, Department of Mechanical Engineering, University of Utah, to be completed May 2002.

Interactions/Transitions

- the PI attended the *IEEE Conference on Decision and Control*, San Diego, CA, Dec. 10-12, 1997.
- the PI attended the *Reconfigurable Flight Control Workshop*, Wright Patterson AFB, OH, Feb. 2-3, 1998, and made a presentation.
- the PI visited Dr. Brent Ellerbroek at the *Starfire Optical Range*, Kirtland AFB, NM, on May 7, 1998. He spent the day discussing problems of adaptive optics with Dr. Brent Ellerbroek and Dr. Don Washburn.
- the PI attended the *Workshop on Dynamic Systems and Control of the Air Force Office of Scientific Research*, Pasadena, CA, May 27-29, 1998.
- the PI attended the *American Control Conference*, Philadelphia, PA, June 24-26, 1998, and made a presentation.
- the PI attended the *AIAA Guidance, Navigation, and Control Conference*, Boston, MA, August 10-12, 1998, .
- the PI delivered software for real-time identification of aircraft parameters to *Boeing* in September 1997, for application to the C-17 transport aircraft. An updated version was sent at the end of August 1998, and also included an application to the Boeing 777. This effort was supported by a separate contract, but capitalized on results obtained under AFOSR sponsorship.
- the PI attended the *International Conference on Noise and Vibration Engineering,* Leuven, Belgium, Sept. 16-18, 1998, and made a presentation.

- the PI attended the *IEEE Conference on Decision and Control*, Tampa, FL, Dec. 16-18, 1998, and made a presentation.
- the PI attended the *World Congress of the International Federation of Automatic Control* (Beijing, China, July 4-9, 1999) and gave part of a course on fault-tolerant control systems.
- the PI attended the *Workshop on Dynamics and Control of the Air Force Office of Scientific Research*, Dayton, OH, August 4-6, 1999, and made a poster presentation.
- John Petersen (graduate student) presented a paper at the *AIAA Guidance, Navigation, and Control Conference*, Portland, OR, August 9-11, 1999. The PI also attended the conference.
- the PI attended the *Restore Final Program Review* (project supported by the Air Force on the subject of reconfigurable flight control), Dayton, OH, August 24-25, 1999.
- the PI attended the *IEEE Conference on Decision and Control*, Phoenix, AZ, Dec. 7-10, 1999, and made a presentation.
- on January 1, 2000, the PI became the Editor-in-Chief of *IEEE Trans. on Control Systems Technology* (http://tcst.elen.utah.edu). The *Transactions* is the main archival journal of the *Institute of Electrical and Electronics Engineers* for the publication of research on the technology and applications of control systems. A substantial fraction of the papers address problems of relevance to the aerospace industry.
- the PI assisted the researchers of *Zona Technology, Inc.*, with their NAVY/SBIR contract: "Adaptive Reconfigurable Control Based on a Reduced Order System Identification for Flutter and Aeroservoelastic Instability Suppression." A meeting was held on Dec. 7, 1999, in Scottsdale, AZ, and the PI also made a presentation at the kick-off meeting of the contract in *Boeing*, St Louis, MO, on Feb. 16, 2000.
- the PI attended the *American Control Conference*, Chicago, IL, June 28-30, 2000.
- the PI attended the *Workshop on Dynamics and Control of the Air Force Office of Scientific Research*, Pasadena, CA, August 21-23, 2000, and made a poster presentation.

FAST IMPLEMENTATION OF DIRECT ALLOCATION WITH EXTENSION TO COPLANAR CONTROLS

John A. M. Petersen and Marc Bodson[*]

Department of Electrical Engineering

University of Utah, Salt Lake City, UT 84112

(801) 581 8590 bodson@ee.utah.edu

Abstract

The paper considers the direct allocation method proposed by Durham. The original method assumed that every three columns of the controls effectiveness matrix were linearly independent. In this paper, the condition is relaxed, so that systems with coplanar controls can be considered. For fast on-line execution, an approach using spherical coordinates is also presented and results of the implementation demonstrate improved performance over a sequential search. Linearized state-space models of a C-17 aircraft and of a tailless aircraft are used in the evaluation.

[*] Marc Bodson is a Professor of Electrical Engineering and a Senior Member, AIAA. John Petersen is a graduate student. Effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grants F49620-97-1-0405 and F49620-98-1-0013. The views and conclusions contained in the paper are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

1. Introduction

In order to increase the reliability of aircraft, configurations with a large number of actuators and control surfaces are advantageous. Reconfigurable control laws may be used to exploit all the available control power despite failures and damages.[1,2] *Control allocation* is the problem of distributing the control requirements among multiple actuators in order to satisfy the desired objectives while accounting for the limited range of the actuators. Although solutions exist for the control allocation problem, an issue of current interest is that of the feasibility of their implementation on existing computers for aircraft with a large number of actuators.[3]

The *direct allocation* approach[4,5,6] is based on the concept of the *attainable moment set* (AMS), which is the set of all the moment vectors that are achievable within the control constraints. The method of direct allocation allows one to achieve 100% of the AMS, whereas some other approaches such as daisy chaining, pseudo-inverse and generalized inverse solutions have been shown to achieve a smaller volume.[7]

In the direct allocation method, the moment vectors are assumed to be related to the controls through the linear transformation $m = CBu$, where m is the resultant moment, u is the set of controls, and CB is referred to as the controls effectiveness matrix. The original method for three-moments developed by Durham was restricted to systems in which any three columns of CB are linearly independent. For this case, the boundary of the AMS consists of parallelograms defined by pairs of controls varying between their limits. As it turns out, the control needed to produce any moment on the boundary of the AMS is unique. In the direct allocation method, moments lying inside the boundary of the AMS are obtained by scaling the controls required to produce a moment of maximum magnitude in the same direction. In a similar manner, moments lying outside the boundary are scaled down to the achievable values. Therefore, controls are always uniquely defined.

If the restriction on CB is not satisfied, then the boundary of the AMS is defined by polygons rather than parallelograms, and each facet is bounded by $2p$ sides, where p is the number of controls defining the polygonal facet. With more than two variables describing the facet, the solution is not always unique, even on the boundary of the AMS. Because this situation occurs when the effects of three or more controls are linearly dependent in a three-dimensional space, the terminology '*coplanar controls*' is introduced to explicitly refer to this case. Systems with coplanar controls are loosely called '*coplanar systems*.' The geometry of the AMS boundary is further described in the paper, and a possible choice for the selection of the control is proposed, given the non-uniqueness properties.

Next, we consider that most of the computational burden in using the direct allocation method lies in finding the facet on the AMS boundary in the direction of the desired moment. Generally, computations may be split into off-line and on-line computations. Off-line computations are defined to be those that may be performed at the design stage or, in the case of a reconfigurable control law, at a slower rate than the normal sampling rate. On-line computations are those that are required for the determination of the control input at every sampling instant. A significant portion of the computations may be performed off-line in the direct allocation method, and consist in the determination of the set of attainable moments. On-line computations include the search for the facet in the attainable moment set that is aligned with the desired moment, and the determination of the control input using appropriate scaling.

To reduce the on-line computations, a representation of the AMS in 2-dimensional space, using spherical coordinates, is shown to be beneficial. The new method converts the AMS representation into a two-dimensional system, where special techniques can be used to accelerate the search. Two options are suggested for the implementation. The first method computes facet boundaries that are used on-line to rapidly eliminate a large number of facets from the search. The second method creates a two-dimensional array relating the spherical coordinates of the desired moment to a corresponding facet identifier. The appropriate facet is found on-line by table look-up, requiring no iterations and virtually no computations. The spherical methods are also developed for coplanar systems. Rather than using polygonal facets for the rapid search, a representation using multiple coplanar sub-facets is considered. Examples used to illustrate the concepts proposed include a C-17 aircraft model with 16 actuators and an advanced tailless fighter model with 11 actuators.

2. Problem Statement

Consider the linearized aircraft model

$$\begin{aligned}\dot{x} &= Ax + Bu \\ y &= Cx\end{aligned} \tag{1}$$

where $x \in R^5$, $u \in R^n$, $y \in R^3$. The states of the aircraft are given by x, and include the angle of attack, the pitch rate, the angle of sideslip, the roll rate, and the yaw rate. The output y contains the pitch rate, the roll rate, and the yaw rate. The control input, u, is constrained to limits

$$u_{i,\min} \le u_i \le u_{i,\max} \qquad \text{for } i = 1 \ldots n$$

The matrix B specifies the forces and moments generated by the actuators. These forces and moments are limited by the allowable range of control inputs. Since we are interested in controlling the output y, we consider the derivative of y, which is given by

$$\dot{y} = CAx + CBu \tag{2}$$

Model reference control laws[8] and dynamic inversion control laws[9] allow one to specify the trajectories of the output of the system by selecting the value of the term CBu due to the control input. The direct allocation problem can be formally stated as follows:

Objective: Given a desired vector m_d, find the vector u such that CBu is closest to m_d in magnitude, with u satisfying the constraints and CBu proportional to m_d.

In the original formulation of Durham, the vector m_d was a desired moment. Here, the vector represents three desired rotational accelerations. We will nevertheless continue to refer to the set of achievable *CBu's* as the AMS.

3. Attainable Moments and Direct Allocation

Initially, we make the following assumption:

Assumption (non-coplanar controls): Every 3x3 sub-matrix of CB is full rank.

Under this assumption, the following properties are obtained.

3.1. Systems with Non-Coplanar Controls

3.1.1. Properties of the AMS

The AMS is a convex polyhedron, whose boundary is the image of the facets of the control space. A facet of the control space is defined as the set obtained by taking all but two controls at their limits, and varying the two 'free' controls within the limits. A 2D facet in control space is rectangular. The projection of such a facet to moment space is a linear transformation resulting in a 2D parallelogram in 3D space. When every set of three columns of the *CB* matrix are linearly independent, every facet on the boundary of the AMS originates from a unique facet on the boundary of the control space. There are $2^{n-2}n!/[2!(n-2)!]$ facets in the control space. However, most of these facets map to the interior of the AMS, and the boundary of the AMS is comprised of only $n(n-1)$ facets[6]. The four corners of each facet of the AMS are called *vertices*, and the four sides are called *edges*. There are $n(n-1)+2$ vertices in the AMS.

3.1.2. Computation of the AMS

The boundary of the AMS is made of facets corresponding to all the possible pairs of input variables. For each pair, there is a multitude of facets in the original control space, but only two of them map to the boundary of the AMS. They may be found by looking for the combination of the other controls that maximizes the distance between the two facets. Hereafter, we refer to one of the facets as a 'max' facet and the other as a 'min' facet. The collection of all these pairs of facets then constitutes the boundary of the AMS.

To further explain the procedure, let *CB* be subdivided as

$$CB = [cb_1\ cb_2 \ldots cb_n],$$

where cb_i is a column vector, and $m_i = cb_i\, u_i$ is the moment vector corresponding to the single control u_i. For a pair of controls, (u_i, u_j), $i \in \{1 \ldots n\}, j \in \{i+1 \ldots n\}$, let the normal to the plane of the facet, η_{ij}, be defined by taking the cross-product of the two vectors defining the facet

$$\eta_{ij} = cb_i \times cb_j \tag{3}$$

Then, the two farthest facets are determined through the two vectors

$$m_{\max} = \sum_{k=1, k\neq i,j}^{n} \mu_{k,\max} \tag{4}$$

$$\text{where} \quad \mu_{k,\max} = \begin{cases} cb_k u_{k,\max} & if\ (cb_k)^T \eta_{ij} > 0 \\ cb_k u_{k,\min} & if\ (cb_k)^T \eta_{ij} < 0 \end{cases}$$

and

$$m_{\min} = \sum_{k=1, k\neq i,j}^{n} \mu_{k,\min} \tag{5}$$

$$\text{where} \quad \mu_{k,\min} = \begin{cases} cb_k u_{k,\max} & if\ (cb_k)^T \eta_{ij} < 0 \\ cb_k u_{k,\min} & if\ (cb_k)^T \eta_{ij} > 0 \end{cases}$$

Note that the case where the vector product $(cb_k)^T \eta_{ij}$ is zero is impossible for non-coplanar systems by virtue of the assumption of linear independence of every three columns of the *CB* matrix. However, this case is possible with coplanar systems and is discussed in detail in section 3.2. In coding this procedure, it is convenient to store an array of flags indicating the control values (max, min, or free) associated to each facet. A facet may also be assigned a number to index the array.

The vertices of the two facets are determined by using the maximum and minimum values of the other two free controls. For instance, the vertices for the max facet are

$$\begin{aligned} v_1 &= m_{\max} + cb_i u_{i,\min} + cb_j u_{j,\min} \\ v_2 &= m_{\max} + cb_i u_{i,\min} + cb_j u_{j,\max} \\ v_3 &= m_{\max} + cb_i u_{i,\max} + cb_j u_{j,\min} \\ v_4 &= m_{\max} + cb_i u_{i,\max} + cb_j u_{j,\max} \end{aligned} \tag{6}$$

Figure 1 shows the results of this procedure for a C-17 aircraft model. A 3-dimensional view of the boundary is shown. The facets are shaded according to height in the yaw acceleration axis. The C-17 model includes sixteen separately controlled surfaces: 4 elevators, 2 ailerons, 2 rudders, and 8 spoilers. The set is delimited by 240 facets.

3.1.3. Computation of the Control Input

The control input is obtained by scaling the desired moment so that the scaled vector reaches the boundary of the AMS. On the boundary, there is a unique relationship between the moment and the value of the input needed to achieve it. If the desired moment is larger than the one attainable in the given direction, the moment vector is scaled to the achievable value. If the desired moment is smaller, the control input associated with the maximum attainable moment is scaled to obtain the desired moment.

The algorithm proceeds as follows. For a given facet, a basis spanning the moment space is formed by using the vector from the origin to one vertex of the facet and the vectors from this vertex to the two adjacent vertices. Let m_{base} be the vector to one of the vertices and Δm_i and Δm_j be the vectors from this vertex to the other two vertices.

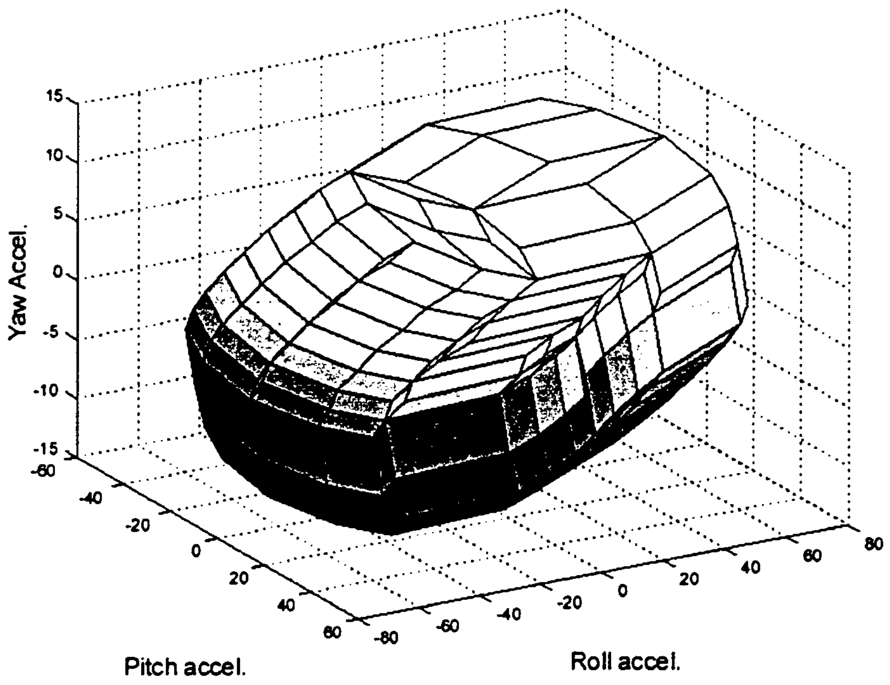

Figure 1. Set of attainable moments for a C-17 model.

Using m_d as the desired moment, we have

$$\begin{aligned}
m_{base} &= v_1 \\
\Delta m_i &= v_3 - v_1 = cb_i \Delta u_i \\
\Delta m_j &= v_2 - v_1 = cb_j \Delta u_j \\
\rho_3 m_d &= \rho_1 \Delta m_i + \rho_2 \Delta m_j + m_{base}
\end{aligned} \tag{7}$$

where $\Delta u_i = \left(u_{i,\max} - u_{i,\min}\right)$. The free parameters ρ_1, ρ_2, ρ_3 are found by solving the 3x3 set of linear equations

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} \Delta m_i & \Delta m_j & m_{base} \end{bmatrix}^{-1} m_d \tag{8a}$$

and letting $\rho_1 = \dfrac{\lambda_1}{\lambda_3}$, $\rho_2 = \dfrac{\lambda_2}{\lambda_3}$, and $\rho_3 = \dfrac{1}{\lambda_3}$. Figure 2 shows graphically the moment vector equation. The value $\rho_3 m_d$ is the point at which the vector m_d intersects the facet. The values of (ρ_1, ρ_2, ρ_3) determine whether m_d intersects the facet. If

$$\begin{Bmatrix} 0 < \rho_1 < 1 \\ 0 < \rho_2 < 1 \\ \rho_3 > 0 \end{Bmatrix} \tag{8b}$$

then a vector in the direction of the desired moment intersects the facet defined by m_{base}, Δm_i, and Δm_j.

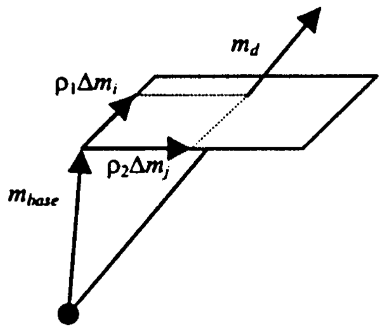

Figure 2. A desired moment intersecting a facet, with basis vectors shown in relation to the facet.

The control vector at the boundary is

$$u_{boundary} = u_{base} + \rho_1 \Delta u_i + \rho_2 \Delta u_j \tag{9}$$

where u_{base}, Δu_i, and Δu_j are the sets of controls which determine m_{base}, Δm_i, and Δm_j respectively. $u_{boundary}$ is the control associated to the maximum moment in the direction of the desired moment and within the control constraints. If $\rho_3 < 1$, the desired moment exceeds the maximum available moment and $u_{boundary}$ is taken to be the control. If $\rho_3 > 1$, the control is scaled to match the moment requirement, with $u = u_{boundary}/\rho_3$.

3.1.4. Sequential Search for Direct Allocation

The computation of the control input involves the solution of a linear system of three equations in three unknowns, and the linear combination of three input vectors. If the correct facet is used, the computations are minor, and the resulting control input satisfies the limits. If the incorrect facet is used, the values of (ρ_1, ρ_2, ρ_3) exceed their limits, and the control input will not satisfy the constraints. The computation may be used as a test of whether the facet is the correct one. If all the facets are tested sequentially in this manner, the procedure may be used for control allocation. We will refer to this approach as the *sequential search* procedure.

The computations for this procedure may be separated into off-line and on-line computations. The off-line code creates a table containing the four vertices associated to each facet. The on-line code consists of retrieving the vertex data, computing the control, and checking its feasibility. Once the correct facet is encountered, computations stop. The search will be time-consuming if the number of facets is large. The sequential search was nevertheless implemented to provide a baseline for the evaluation of the benefits of the methods proposed. More intelligent search techniques have been proposed[5,10], but these were not implemented for this paper. Instead, the use of spherical coordinates is investigated to accelerate the search.

3.2. Systems with Coplanar Controls

3.2.1. Properties of the AMS

For systems with coplanar controls, a p-dimensional volume ($p \geq 2$) in control space maps into a 2D facet in moment space and has $2p$ sides. The facet becomes a polygon defined by p controls. Figure 3 shows the AMS for an advanced tailless fighter model[11]. According to reference (11),

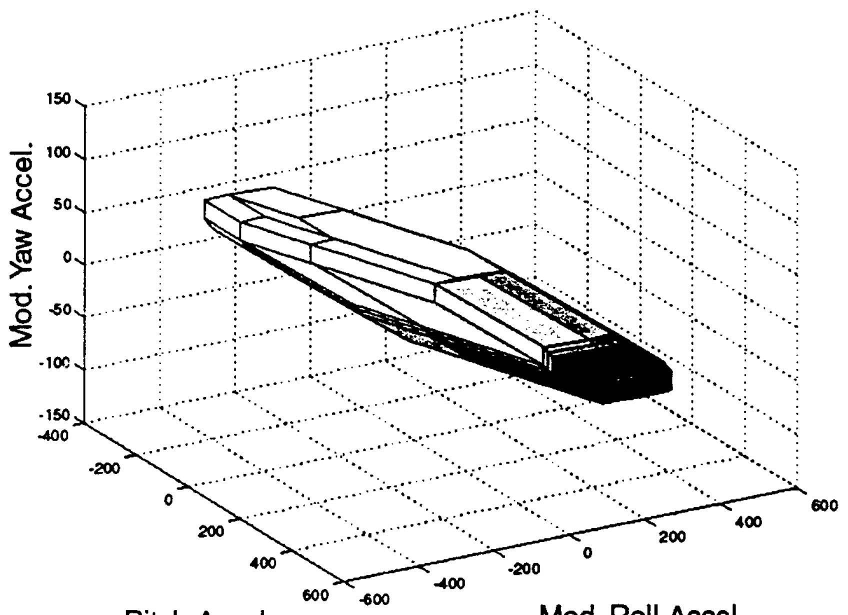

Figure 3: Set of attainable moments for an advanced tailless fighter model.

the output vector y is composed of modified rotational rates. Specifically, the components of y are the pitch rate, the stability axis roll rate, and a blend of sideslip and stability axis yaw rate.

The advanced tailless fighter model includes eleven separately controlled surfaces consisting of elevons, pitch flaps, thrust vectoring, outboard leading edge flaps, spoiler slot deflectors and all-moving tips. In this model, pitch thrust vectoring and the pitch flaps produce linearly dependent moments yielding coplanar controls with any third control variable. It was found that up to four control variables were coplanar $(2 \le p \le 4)$. Some polygonal facets are indeed clearly visible on the figure. The boundary of the AMS is delimited by 78 such facets.

Polygonal facets can also be described by a set of sub-facets that are the projections of the 2D facets of the control space. Each 2D facet is determined by two controls as in the non-coplanar case. For every pair of the p controls, there are 2^{p-2} identical sub-facets offset from each other in the same plane. Since there are $p(p-1)/2$ pairs of controls, the total number of sub-facets covering a polygonal facet is given by

$$n_{pf} = p(p-1)2^{p-3} \tag{10}$$

Figure 4 is a series of figures that details the relationship between a polygonal facet and its sub-

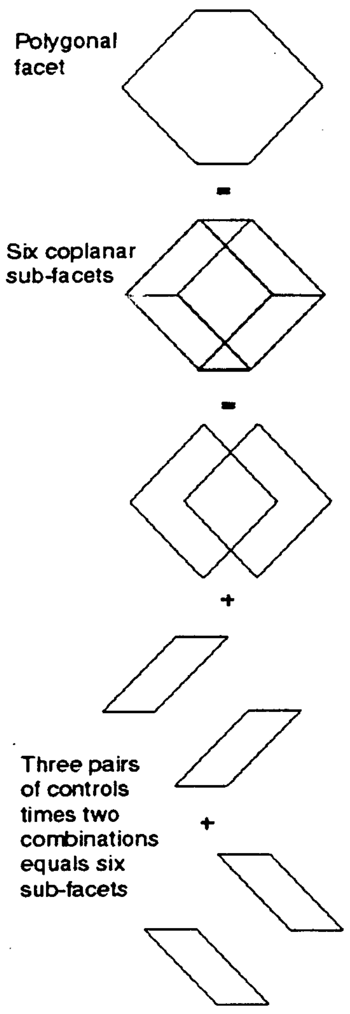
Polygonal facet
=
Six coplanar sub-facets
=
+
Three pairs of controls times two combinations equals six sub-facets
+

Figure 4. View of the relationship between a parent polygonal facet and its sub-facets.

facets. The top figure shows a polygonal facet. The middle figure shows how the six sub-facets cover the polygonal facet. The bottom figure shows the three sets of sub-facets in an exploded view. The representation of the AMS by sub-facets has been found to be practical for the computation of the control input in the extension of the direct allocation method proposed here.

3.2.2. Computation of the AMS

The algorithm is similar to the one described in section 3.1.2. For every pair of controls, (u_i, u_j), $i \in \{1 \ldots n\}$, $j \in \{i+1 \ldots n\}$, each column of *CB*, excluding columns *i* and *j*, is evaluated. Coplanar controls can be determined by the product $(cb_k)^T \eta_{ij}$ where $k \neq i,j$. If this product is zero, then the k^{th} control is coplanar with the facet created by the control pair (u_i, u_j). The non-coplanar controls (that correspond to the columns of *CB* that are linearly independent of columns *i* and *j*) are then selected so as to maximize the distance of the facet from the origin. As before, for each facet, there is an identical facet that lies on the opposite side of the AMS boundary and can be found using the opposite values for each of the *n-p* non-coplanar controls. Defining the set of indices

$$K = \left\{ \begin{matrix} i, j, \text{ and indices of all controls} \\ \text{coplanar with } u_i \text{ and } u_j \end{matrix} \right\},$$

the maximum displacement vector is

$$m_{\max} = \sum_{\substack{k=1 \\ k \notin K}}^{n} \mu_{k,\max} \tag{11}$$

where $\mu_{k,\max}$ is defined in (4). The opposite facet is determined by the minimum displacement vector

$$m_{\min} = \sum_{\substack{k=1 \\ k \notin K}}^{n} \mu_{k,\min} \tag{12}$$

where $\mu_{k,\min}$ is defined in (5).

While the non-coplanar *n-p* controls determine the distance of a polygonal facet from the origin, the remaining *p* controls determine the shape of the polygonal facet. Each polygonal facet is made up of $\frac{p(p-1)}{2}$ sets of $r = 2^{p-2}$ sub-facets. Each sub-facet lies in the same plane, but has a different offset that shifts it with respect to the other sub-facets. The offset is determined by the sum of the *r* combinations of max and min control values of the coplanar controls. For every unordered pair of controls $\{(u_a, u_b) \mid a \in K, b \in K, a \neq b\}$, the offset of each sub-facet is computed using the controls $\{u_c \mid c \in K, c \notin \{a,b\}\}$ in different combinations of their upper and lower limits, resulting in

$$offset_q = cb_c u_c \quad \text{where } q \in \{1 \ldots r\}. \tag{13}$$

Since c is a vector of p-2 indices, cb_c in (13) is a matrix with p-2 columns.

Finally, sub-facets are defined by vertices obtained by summing the displacement vector, the offset, and the four combinations of maximum and minimum values of the two free controls, u_a and u_b. For instance, the four vertices for the q^{th} max sub-facet are

$$\begin{aligned}
v_{1,q} &= m_{\max} + cb_a u_{a,\min} + cb_b u_{b,\min} + offset_q \\
v_{2,q} &= m_{\max} + cb_a u_{a,\min} + cb_b u_{b,\max} + offset_q \\
v_{3,q} &= m_{\max} + cb_a u_{a,\max} + cb_b u_{b,\min} + offset_q \\
v_{4,q} &= m_{\max} + cb_a u_{a,\max} + cb_b u_{b,\max} + offset_q
\end{aligned} \tag{14}$$

<u>3.2.3. Computation of the Control Input</u>

With all but two of the controls at their limits for a given sub-facet, the control that will achieve a moment vector intersecting the sub-facet can be defined as before. However, since overlapping sub-facets may exist at a particular boundary point, there is not, in general, a unique relationship between the moment and the value of the input needed to achieve it. While the solution corresponding to any sub-facet could be taken as a solution to the direct allocation problem, we propose to take instead the average of the inputs resulting from all overlapping sub-facets. Taking the average is a simple solution that gives the desired moment, and usually reduces the number of saturated controls, as well as guarantees the continuity of the solution. Figure 5 helps one visualize the advantage of averaging the controls. The figure shows two sub-facets in control space that overlap in moment space. Any point along the dashed line will result in the desired moment, but the mid-point will offer a control vector that yields an attractive compromise between control limits.

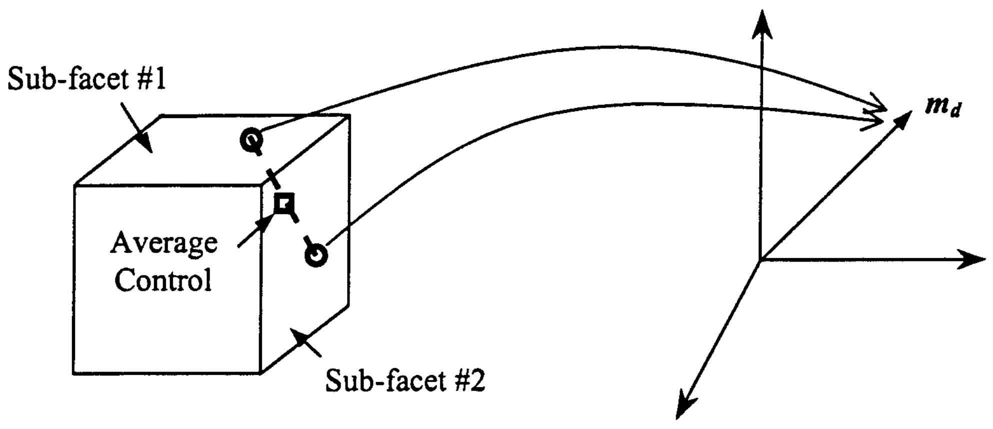

Figure 5. Visualization of averaging the controls

3.2.4. Sequential Search for Direct Allocation

The sequential search procedure described in section 3.1.4 may be employed for the search for the right sub-facet. Although all sub-facets containing the desired moment must be found, once a correct sub-facet is encountered, one only needs to check the other sub-facets lying in the same plane (*i.e.* those with the same parent polygonal facet) to complete the search. The sequential search procedure is useful as a baseline for evaluation.

4. Rapid Search Using Spherical Coordinates

4.1. Systems with Non-Coplanar Controls

4.1.1. Representation of the AMS in Spherical Coordinates

Because the determination of the applicable facet does not depend on the magnitude of the desired moment, the search may be performed in a 2-dimensional space instead of the original 3-dimensional space. Each vertex of the moment space, determined by (x, y, z) coordinates, can be expressed in spherical coordinates (θ, $s\varphi$, ρ), with

$$\theta = \tan^{-1}(y\,,\,x) \tag{15}$$

$$s\varphi = \sin(\varphi) = \frac{z}{\rho} \tag{16}$$

$$\rho = \sqrt{x^2 + y^2 + z^2} \tag{17}$$

θ represents the *azimuth angle* (the horizontal angle in the x, y plane), φ represents the *elevation angle* (the vertical angle from the x, y plane), and ρ represents the distance from the origin. This third spherical coordinate is irrelevant for the search of the facet. For the azimuth, note that a two-argument inverse tangent function is used. The value $\sin(\varphi)$ (henceforth abbreviated $s\varphi$) is also used instead of φ to simplify on-line computations.

Figure 6 shows the result of transforming the boundary of the C-17 AMS to spherical coordinates, with θ shown on the x-axis in the range of $\pm\pi$. The sine of the elevation angle is shown on the y-axis. The figure shows that the lines that form the edges of the facets become curves, because of the nonlinear change of coordinates. In fact, these curves are the well-known *great circles* used in navigation. They are the projection on the unit sphere of 3D line segments, or the intersection of the unit sphere with a plane including the origin and the two vertices. The idea of using the spherical coordinates is that the desired moment is represented in the 2D space as a point, and that the control allocation problem becomes the simpler problem of determining to which 2D facet the point belongs.

Some terminology is introduced to help convey the algorithms. An *edge* joins two adjacent vertices of a facet. A *crossing* is defined as an edge between two vertices, with azimuth angles θ_1

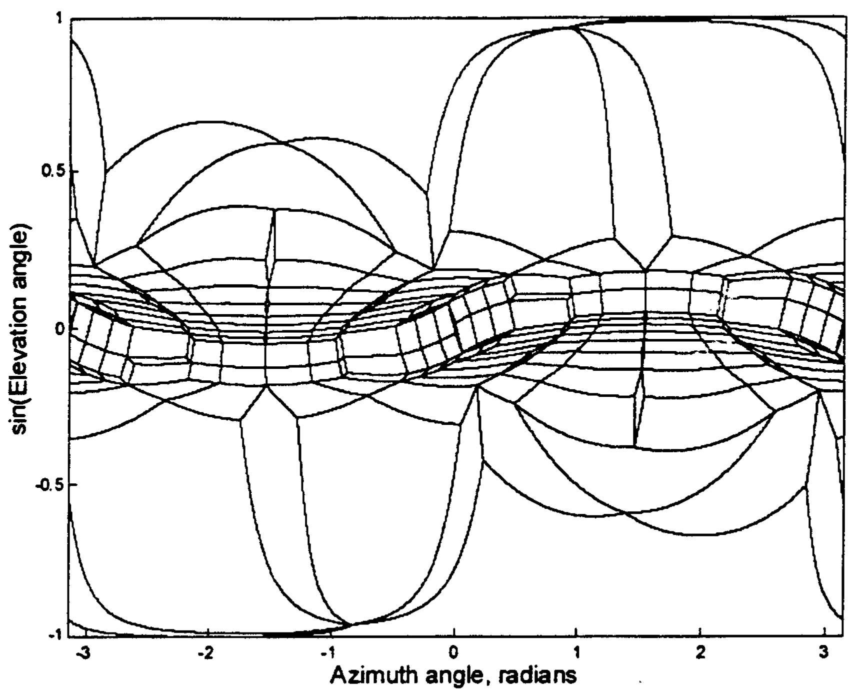

Figure 6. Spherical mapping of C-17 AMS facets.

and θ_2 crossing the $\pm\pi$ boundary. Existence of a crossing may be determined by testing

$$\delta\theta = \theta_{max} - \theta_{min}, \tag{18}$$

with $\theta_{max} = \max(\theta_1, \theta_2)$, $\theta_{min} = \min(\theta_1, \theta_2)$. If $\delta\theta > \pi$, one may conclude that the edge crosses the boundary. A *split facet* is one in which the facet is bisected by the line $\theta = \pm\pi$.

The great circle for each pair of vertices looks like a distorted sinusoid in the spherical coordinate space. The curve reaches maximum and minimum values of the elevation angle φ that have equal magnitude and opposite sign. These points occur 180° apart in azimuth angle. For the mapping of two vertices in 3D to two vertices in spherical coordinate space $\{(x_1,y_1,z_1),(x_2,y_2,z_2)\} \rightarrow \{(\theta_1, s\varphi_1),(\theta_2, s\varphi_2)\}$, it turns out that one of the extrema of the great circle occurs at $(\theta_{pk}, s\varphi_{pk})$ given by

$$\theta_{pk} = \tan^{-1}(x_1z_2 - x_2z_1,\ y_2z_1 - y_1z_2) \tag{19}$$

$$s\varphi_{pk} = \frac{s\varphi_1}{\sqrt{s\varphi_1^2 + (1 - s\varphi_1^2)(\cos(\theta_{pk} - \theta_1))^2}} \tag{20}$$

To obtain the correct values, θ_{pk} needs to be adjusted by $\pm\pi$ using the following rule:

$$\textit{if} \quad \cos\left(\theta_{pk}-\theta_1\right)<0, \quad \textit{then} \;\; \theta_{pk}=\theta_{pk}-sign\left(\theta_{pk}\right)\pi \tag{21}$$

The other extremum of the great circle is obtained by symmetry.

With the knowledge of the peaks of the great circle, the equations defining the great circle, that is, the edge of the facet under consideration, is given by

$$\begin{aligned}
&\cos\left(\varphi_{pk}\right)=1-s\varphi_{pk}^2\\
&\alpha=s\varphi_{pk}\cos\left(\theta_{pk}-\theta_k\right)\\
&s\varphi_k=\frac{\alpha}{\sqrt{\cos(\varphi_{pk})+\alpha^2}}
\end{aligned} \tag{22}$$

where $(\theta_k, s\varphi_k)$ is a point on the great circle.

4.1.2. Rapid Search using Facet Boxes

In the first option, ranges are computed for the coordinates of the facets, and they define boxes in which the facets are located. The boxes are used to quickly assess whether the desired moment is likely to lie within a given facet. The overall approach is similar to the exhaustive search, but simple inequality tests are used to drop facets from the list. For those facets that are left, the usual test of (8b) is performed. If the test is successful, the control input is quickly obtained. Otherwise, the search continues. The idea is that the test is required for very few facets.

4.1.2.1. Off-line Computations

Off-line computations consist in the determination of the AMS, and of the boxes that delimit the facets in spherical coordinates. For illustration, a facet box is outlined with a dashed line in figure 7. It shows that the box is determined not only by the coordinates of the vertices, but also by maxima reached within the edges of a facet. The peaks of the great circles are therefore determined using (19), (20), and (21), and their values are used in the computations of the box if the peaks lie between the vertices.

A difficulty with the implementation of the method is that facets may span the boundaries of the 2D space. In particular, two facets include the *north pole* and the *south pole*. The north and

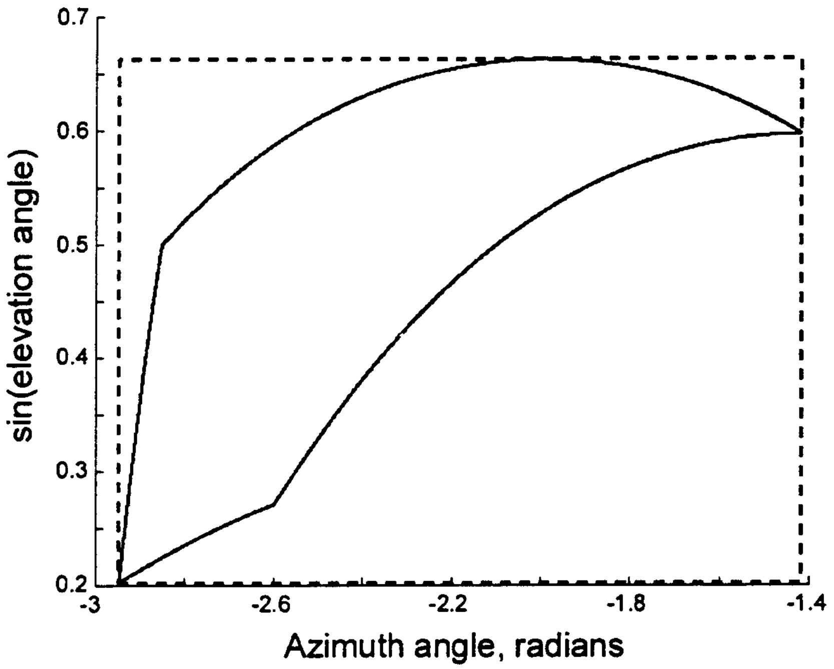

Figure 7. Spherical mapping of a single facet outlined by a facet box.

south poles are the points with $\varphi = 90^{\circ}$ and $\varphi = -90^{\circ}$, respectively. Facets may also span the azimuth boundaries. Such facets could be split into two facets to perform box tests. Instead, however, facet types are defined, and those facets that span the azimuth boundary are redefined on the $(0, 2\pi)$ range so that they become contiguous. The procedure is then implemented as follows.

Step 1: *Compute AMS vertex coordinates and facets.* The AMS is computed using the direct method as described above.

Step 2: *Compute spherical coordinates of all vertices.*

Step 3: *Determine the facet type.* Five types of facets are considered and are designated as types 0, 1, 2, 3, and 4. The azimuth range is extended so that each facet is completely contained in at least one of the two ranges of θ: $-\pi < \theta < \pi$ and $0 < \theta < 2\pi$. A facet is assigned a label of type 0 (those facets in the first set) or type 2 (those facets in the second set, *i.e.* split facets). Further, three special cases are assigned: (a) facets that enclose a pole (type 1), (b) facets that border a pole and lie within $-\pi < \theta < \pi$ (type 3), and (c) facets that border a pole and lie within the range $0 < \theta < 2\pi$ (type 4). Facet type can be determined by testing the $\delta\theta$ of each facet edge. The value of $\delta\theta$ is categorized in four possible ranges:

$$\textit{Case 1}:\ \delta\theta < \pi,\quad \textit{Case 2}:\ \delta\theta > \pi,\quad \textit{Case 3}:\ \delta\theta = \pi,\quad \textit{Case 4}:\ \delta\theta = 0.$$

A simple algorithm can be applied to determine the facet type based on the type and number of crossings, with

$$\textit{facet type} = c + 3d \tag{23}$$

where $c \in \{0,1,2\}$ is the number of crossings and $d \in \{0,1\}$ is the number of occurrences of case 3.

Step 4: *Determine box boundaries of spherical facets.*

(a) *Compute the maximum and minimum spherical coordinates of the edges between vertices of each facet.* If the facet is type 2 or type 4 (*i.e.*, in the $0 < \theta < 2\pi$ range), negative values of θ_1, θ_2, and θ_{pk} of each edge are incremented by 2π before calculating $s\varphi_{pk}$.

(b) *Store the extremal values of θ and sin(φ) for each facet to define the facet box.* For each facet, determine θ_{min}, θ_{max}, $s\varphi_{min}$, and $s\varphi_{max}$. Store these values in a facet box table. If the facet includes a north (south) pole, $s\varphi_{max}$ ($s\varphi_{min}$) is forced to its maximum (minimum) of 1 (-1). Distinguishing a north pole from a south pole can be done by calculating the great circle formed by any two of the facet vertices that form an edge and testing the location of a third facet vertex (θ_3, $s\varphi_3$) relative to this great circle. If $s\varphi_{gc}$ is computed from (22) with $\theta_k = \theta_3$, and if $s\varphi_3 > s\varphi_{gc}$, the facet is a north pole. Otherwise, it is a south pole.

Step 5: *Compute and store the 3x3 inverse of equation (8a) for each facet.*

4.1.2.2. On-line Computations

Step 1: *Convert the desired moment into spherical coordinates.*

Step 2: *For each facet, check the feasibility of the desired moment.* Compare the coordinate of the desired moment to the facet box. If the facet box is type 2 or type 4, add 2π to the azimuth of the point. If the desired moment lies within the box boundaries, compute (ρ_1, ρ_2, ρ_3) as shown in section 3.1.3. The rest of the controls are given by the control flags associated with the facet number.

Step 3: *Compute the control input.* Once the correct facet is found and the test in (8b) is performed, one only needs to scale the control as necessary to satisfy the constraints.

4.1.3. Rapid Search using Table Look-up

The second option consists in creating a look-up table $f(\theta, s\varphi)$ which gives the number of the facet associated with a given pair of spherical coordinates. If the azimuth and elevation angles are quantized with 1000 points each, this option requires an array with 1,000,000 values, a size which is large but within the reach of existing computers. The creation of the table is essentially the transcription of figure 6 into an array, and the marking of the elements of the array with the associated facet number.

The on-line computations could not be simpler with this approach: the facet towards which the desired moment points is found instantly by table look-up, and the appropriate control is determined with minor computations. Note that control allocation is guaranteed to be performed within a known and short period of time.

4.1.3.1. Off-line Computations: Construction of the Facet Table

The steps to this method are as follows.

Step 1: *Compute AMS vertex coordinates and facets.* The AMS is computed using the direct method as before. From this computation, one obtains coordinate information as well as knowledge of which vertices connect along an edge of the facet, and the set of controls that form each facet.

Step 2: *Compute spherical coordinates of the vertices.* For a vertex at (x, y, z), the spherical coordinates $(\theta, s\varphi, \rho)$ are obtained using eqs. (15, 16, and 17).

Step 3: *Compute and quantize the facet edges.* Quantizing the edges is done by converting the end points to a range of index values in θ, computing corresponding values of $s\varphi$, and then converting these values to an appropriate index. Continuous edges in the range $-\pi<\theta<\pi$ are straightforward. Edges which cross the $\theta = \pm\pi$ boundary have $s\varphi$ indices that are not contiguous and must be managed properly. The indices of each edge of the facet are stored in a common array.

Step 4: *Modify pole facets.* Facets containing a pole must be treated as special cases. A facet with one crossing, that is, one edge with $\delta\theta = \pi$ (corresponding to an edge that passes through a pole), contains a pole. Since the line $s\varphi$ is actually a single point in 3D, the edge $s\varphi = \pm 1$ (1 for north pole, -1 for south pole) for each θ index must be added to the common array of step 3.

Step 5: *Create facet table.* The facet table is a two-dimensional array of the facet numbers or identifiers. The indices represent quantized values of θ and $s\varphi$. When the common array is complete, the facet table is updated in the storage locations to which the facet indices just calculated correspond, including indices inside the boundary of the facet.

Step 6: *Compute and store the 3x3 inverse of equation (8a) for each facet.*

4.1.3.2. On-line Computations

Step 1: *Convert the desired moment into spherical coordinates and then to facet table indices.*

Step 2: *Obtain the facet number from the look-up table.*

Step 3: *Compute the control input.* Compute (ρ_1, ρ_2, ρ_3) as shown in section 3.1.3 to arrive at the values for the free controls. The rest of the controls are given by the control flags associated with the facet number. Scale the control as necessary to satisfy the constraints.

4.1.4. C-17 Example

Each algorithm was tested with 1000 randomly selected desired moments for the C-17 example. The AMS has 240 facets and 242 vertices in moment space. The sequential search method was used to establish a baseline to evaluate the other search methods. The spherical facet table technique was simulated using quantizations of 100 and 1000 for each axis.

Figures 8 and 9 display comparisons of the number of floating point operations (obtained by using the *flops* command in MATLAB®) of the three algorithms. These histograms are intended only to provide a rough comparison of the algorithms described in this paper and should be used with caution, as the results are dependent on the specific implementation as well as on the language and hardware used. Note in particular that MATLAB® counts as floating pointing

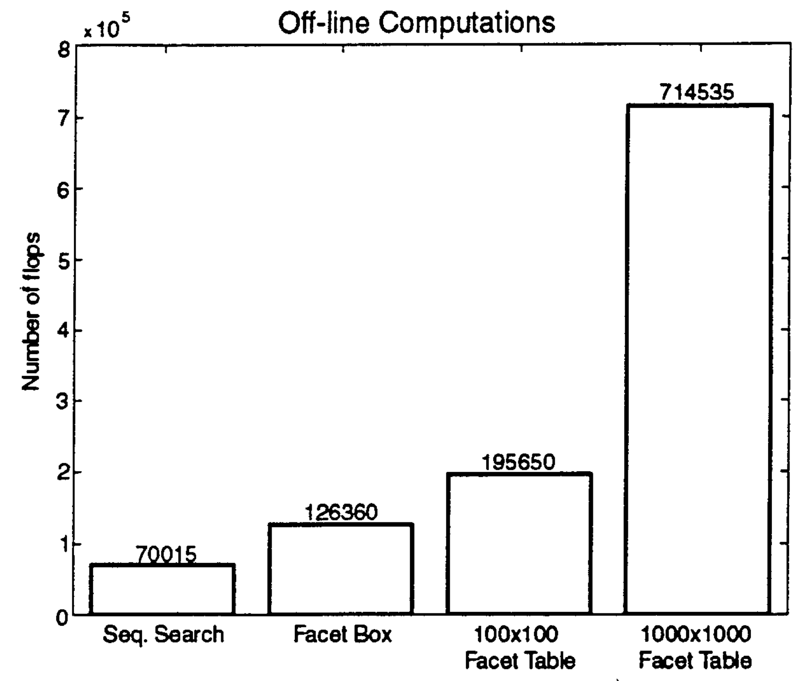

Figure 8. Histogram showing comparison of off-line computations for the different methods.

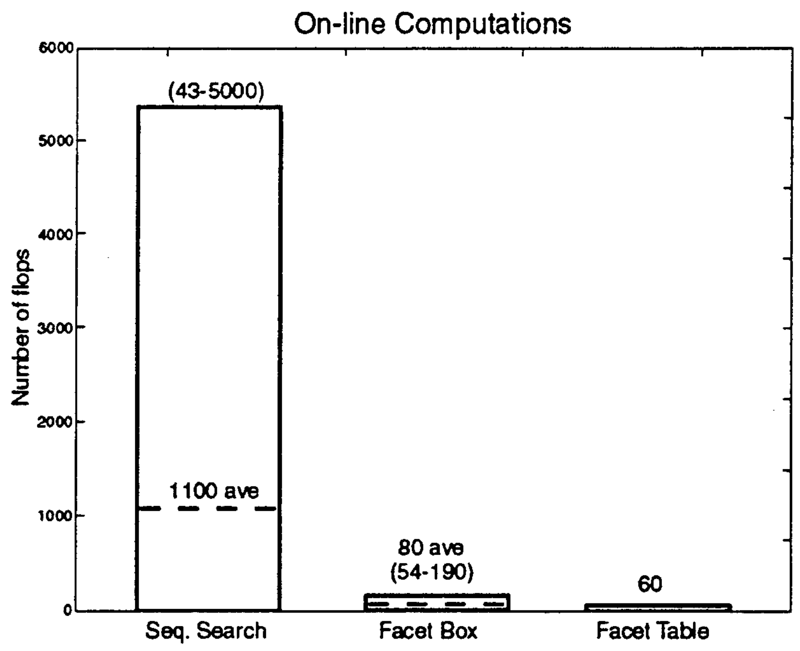

Figure 9. Histogram showing comparison of on-line computations for the different methods.

operations some operations that would normally be counted as integer operations. The code for each algorithm was written in MATLAB® version 5.3.

The histogram in figure 9 shows a range and an average value for the on-line code of the sequential search and facet box methods. The range denotes the minimum to maximum possible values for the method, whereas the average was computed for the 1000 randomly sampled moments. Our simulations have shown that a sample size of 1000 or more is typically large enough for the average numbers to be representative. One finds that the facet table approach considerably reduces the number of required computations to be performed on-line, and eliminates the variability in the number of those computations, making it ideal for real-time applications that demand reliable and predictable results. Memory requirements are significant, however. For memory rich systems requiring extremely fast operation, such as found in computer simulations, this approach is an attractive option. In an adaptive control application, the off-line computations may also constitute an important burden to be considered. The facet box approach is a simple and useful intermediate option. The facet boxes may also be arranged using a hierarchical tree structure as is commonly done in range search techniques.[12] This may be advantageous for systems with a very large number of controls, but for systems with $n \leq 20$, the reduction in on-line processing is minimal and not considered advantageous enough to warrant its implementation.

Further analysis indicates some interesting characteristics of the facet box algorithm. Figure 10 gives a histogram of the number of box tests performed before the correct facet is found. The theoretical maximum is 240 in this example, but one finds that rarely more than 150 tests are required. The average number of tests was computed to be 49.7.

With the box test, the number of box tests passed is considerably less than the number of facets tested. Figure 11 shows that in nearly 50% of the cases, only one facet passed the box test. In 95% of the cases, a maximum of three facets passed box tests, and in none of the cases did more than five pass the test. The average number of for which (8) must be computed was found to be 2.0, to be compared with 49.7 required without the box test. Although the sequential search is viable, box tests in spherical coordinates make this approach much faster. The number of computations required, however, is not fixed.

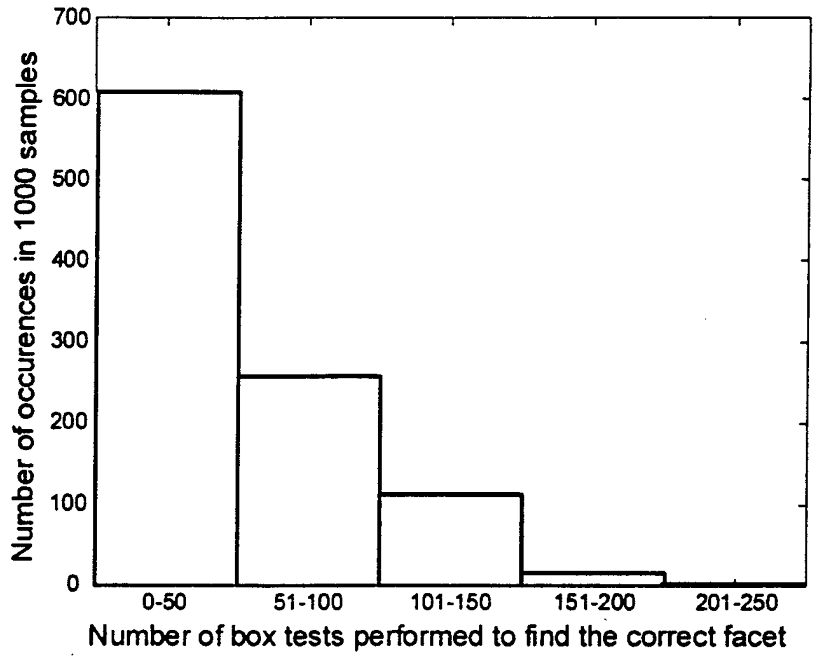

Figure 10. Histogram of facet box tests performed.

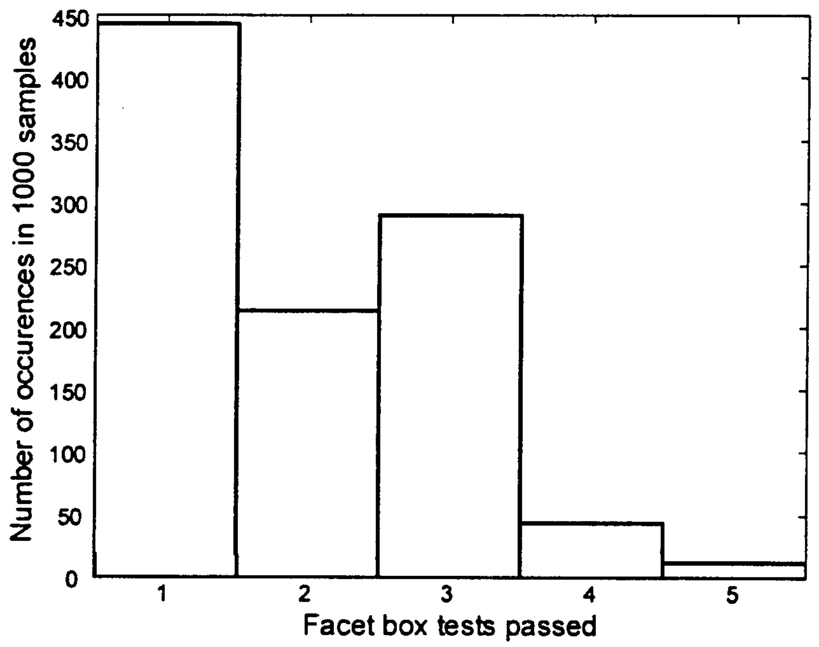

Figure 11. Histogram of the facet box tests passed before finding the right facet.

4.2. Systems with Coplanar Controls

4.2.1. Representation of the AMS in Spherical Coordinates

Figure 12 shows the result of transforming the boundary of the advanced tailless fighter AMS to spherical coordinates. The transformed polygonal facets are visible. Although not shown, parallelogram sub-facets are used to cover the area of every polygonal facet. Because the polygonal facets can be completely specified by sub-facets, the same options as discussed in sections 4.1.2 and 4.1.3 can be used. However, modifications to the specific options are made here to extend its use to solve the problem of overlapping sub-facets.

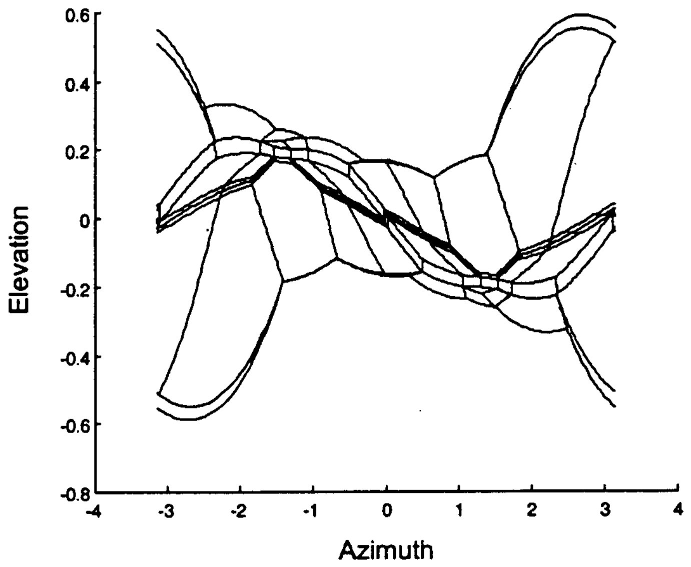

Figure 12. Spherical mapping of overlapping facets outlined by facet boxes.

4.2.2. Rapid Search using Facet Boxes

4.2.2.1. Off-line Computations

As in the original method, ranges are computed for the coordinates of the sub-facets. Sub-facets are treated no different from non-coplanar facets. However, a methodology for tracking overlapping sub-facets improves the on-line search. An array identifies a parent polygonal facet for each sub-facet. Once a sub-facet is found, only the other sub-facets associated with the same parent are tested. For our example of the tailless fighter, which has 78 polygonal facets and 210 sub-facets, the array has 210 rows and 78 identifiers. Figure 13 shows an illustration of overlapping sub-facets outlined with facet boxes.

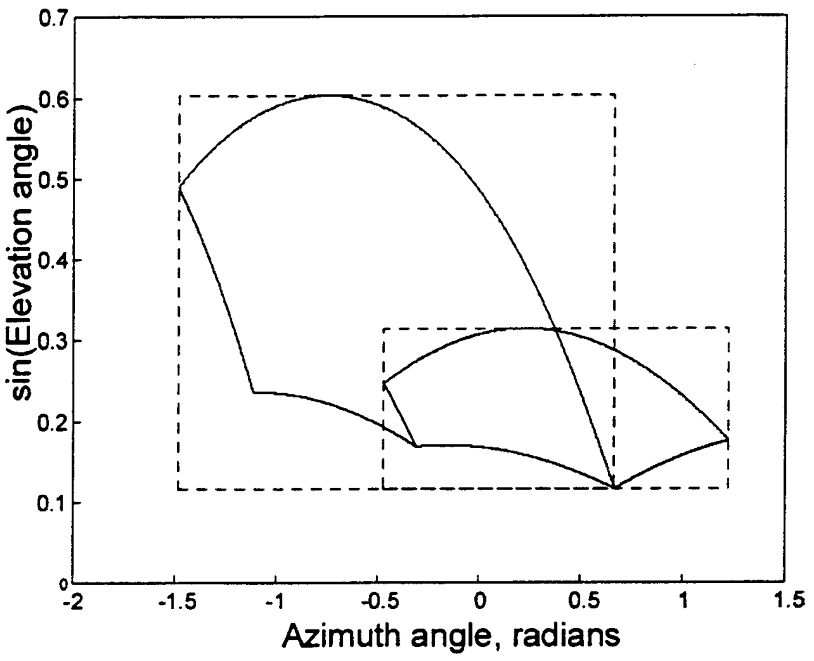

Figure 13: Spherical mapping of overlapping sub-facets outlined by facet boxes.

<u>4.2.2.2. On-line Computations</u>

<u>Step 1:</u> *Convert the desired moment into spherical coordinates.*

<u>Step 2:</u> *For each sub-facet, check the feasibility of the desired moment.* Compare the coordinate of the desired moment to the facet box. If the desired moment lies within the box boundaries, compute the control for this sub-facet. If the conditions on (ρ_1, ρ_2, ρ_3) are satisfied, check all other sub-facets associated with the same parent facet. If not, go to the next sub-facet.

<u>Step 3:</u> *Compute the control input.* Once the correct sub-facets are found and the tests of (8b) are performed, the applicable controls are averaged, and scaled if the desired moment exceeds the achievable value.

<u>4.2.3. Rapid Search using Table Look-up</u>

The creation of a spherical coordinate table for a system with coplanar controls can be done by including a third dimension. The added dimension is a vector of n_{pf} values corresponding to the sub-facets that overlap at that location. Low quantization resolution may increase the apparent number of overlapping sub-facets.

4.2.3.1. Off-line Computations: Construction of the Facet Table

The facet table is a two-dimensional array of sub-facet numbers or identifiers. Sub-facet indices are recorded along a third dimension, if overlapping occurs. Again, the details of this method are identical to those explained in section 4.1.3 with the modification of the vectorized facet identifiers in the table.

4.2.3.2. On-line Computations

Step 1: *Convert the desired moment into spherical coordinates and then to indices of the facet table.*

Step 2: *Obtain the array of facet numbers from the look-up table.*

Step 3: *Compute the control input.* Compute (8) for each sub-facet in the array to arrive at the values for the free controls. Average the resulting controls of each sub-facet. Scale the averaged control as necessary.

Again, the on-line computations are quite simple with the facet table approach: the sub-facets towards which the desired moment points are found instantly by table look-up, and the appropriate control is determined with minor computations. Note that on-line computations could be even further reduced by converting the facet table into a *control* table. This would be done by computing the control for each facet table location in the off-line code. The on-line code then becomes exclusively a look-up table to obtain the control. This method is ideal for applications where low resolution is acceptable or where simple and reliable on-line code is of premium importance.

4.2.4. Tailless Fighter Example

Each algorithm was tested with 1000 randomly selected desired moments and with the tailless fighter model. The AMS has 210 sub-facets in moment space. The sequential search method was used to establish a baseline to evaluate the other search methods. The spherical facet table technique was implemented using quantizations of 100 and 1000 for each axis. Figure 14 displays a comparison of floating point operations for the off-line computations. The number of computations for the 1000x1000 facet table was of the order of 10^7 flops and was not plotted with the others in the histograms.

Figure 15 shows the on-line computations. The histogram entries for the sequential search and facet box methods have a range and an average value. The range denotes the minimum to maximum possible values for the method, whereas the average was computed for the 1000 randomly sampled moments. As in the case for systems with non-coplanar controls, both spherical approaches considerably reduce the number of computations to be performed on-line, and significantly reduces the variability in the number of those computations. The facet box approach is nearly equivalent to the facet table method in terms of on-line calculations and significantly cheaper in terms of off-line computations.

The variation in on-line computations for the facet table method of different quantizations is due to variations in the number of apparent overlapping sub-facets. Although two sub-facets might not overlap, if the quantization is low enough, they effectively may. This results in more sub-facets to be checked and averaged. Therefore a higher quantization will typically yield slightly lower on-line computations.

Although the maximum number of box tests is 210, the average number is only 52. Figure 16 is a histogram of the number of box tests performed before the correct facet is found. Figure 17 shows a histogram of the number of box tests performed before a final control value is determined. In 92% of the cases, a maximum of eight facets passed the test, and in none of the cases did more than fifteen pass. The average number of tests of (8b) was computed to be 5.6, to be compared with 52 required without the box test. The number of computations required, however, is not fixed due to variability in the number of overlapping facets.

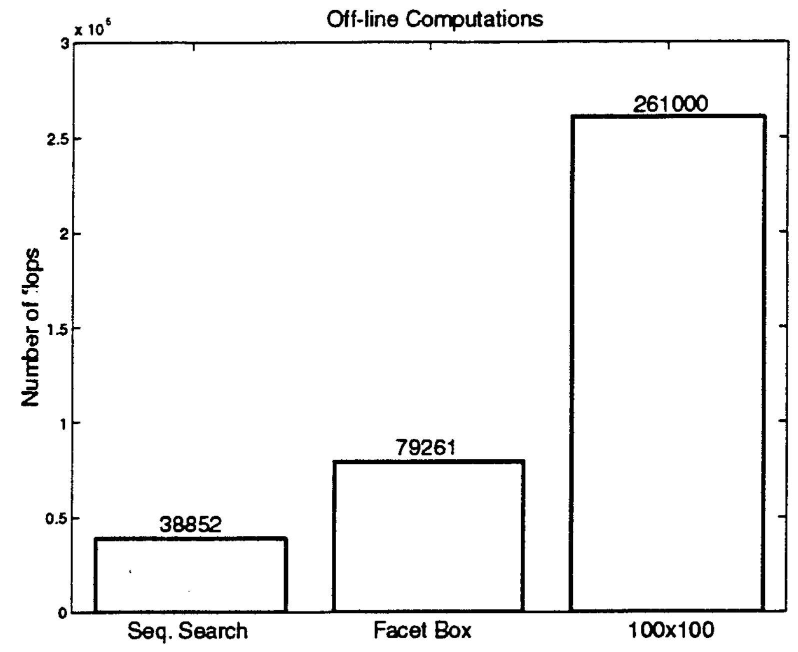

Figure 14. Histogram showing comparison of off-line computations for direct allocation methods for an 11 actuator coplanar system.

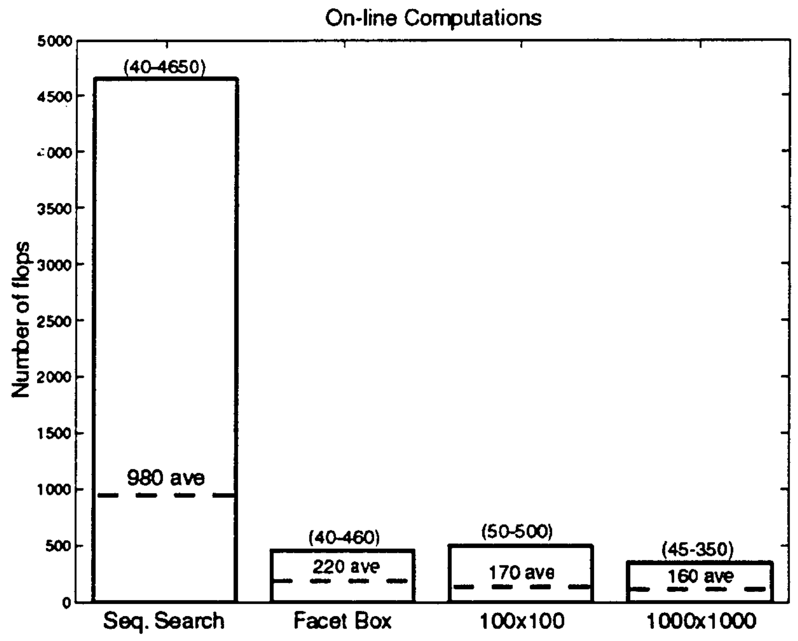

Figure 15. Histogram showing comparison of on-line computations for direct allocation methods for an 11 actuator coplanar system.

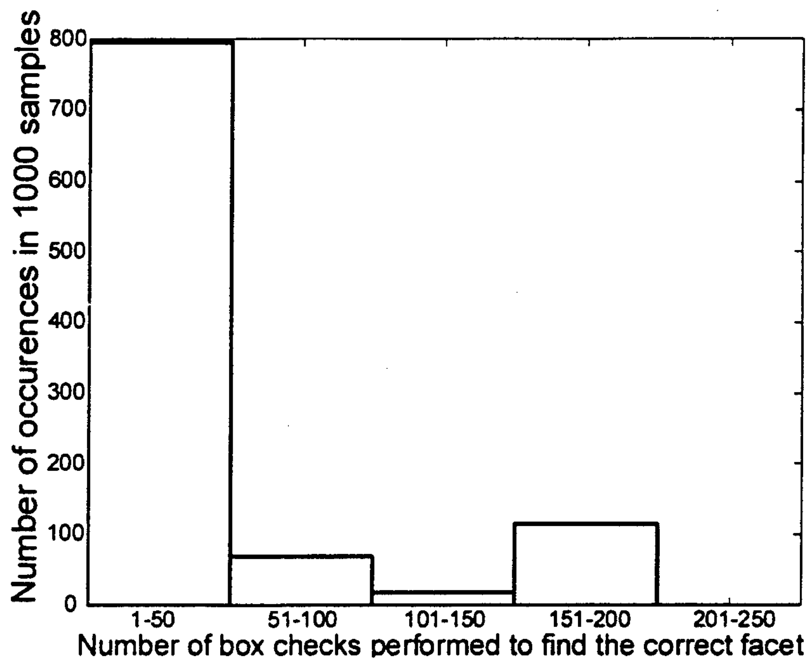

Figure 16. Histogram of sub-facets checked.

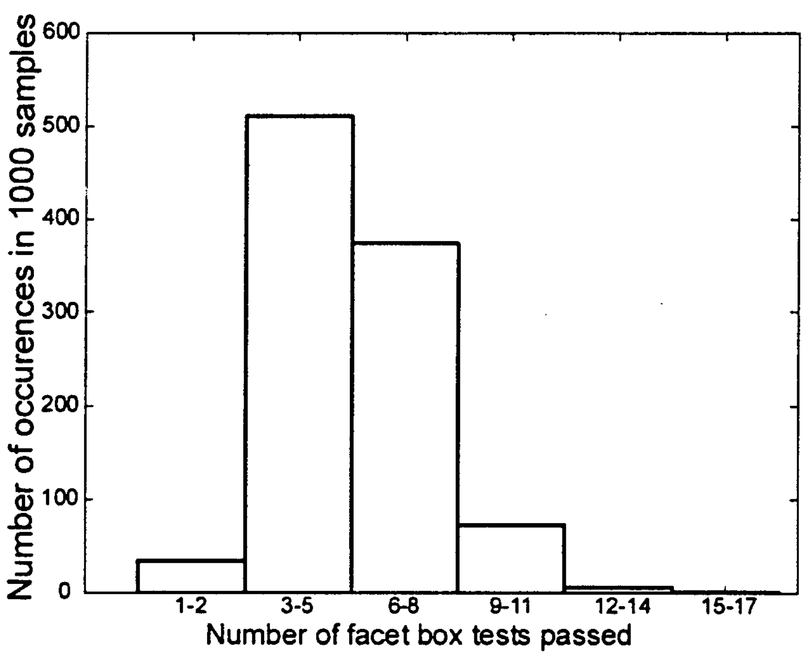

Figure 17. Histogram of the number of facet box tests passed before finding the correct facet.

5. Conclusions

Direct allocation provides a solution to the control allocation problem that not only retains the direction of the desired moment, but takes advantage of the maximum attainable moment set. Direct allocation was previously only applicable to systems whose controls effectiveness matrix was such that every three columns were linearly independent. A system that is not limited in this way has been called a system with coplanar controls, or simply a coplanar system. The example of a tailless fighter aircraft model was shown to fall in this category.

The geometry of the attainable moment set for a coplanar system was explained and an extension to the direct allocation method was given in this paper. The average of the multiple solutions was computed in the procedure, and the concept of overlapping sub-facets was found useful for that purpose. Although the method of direct allocation is stressed, the shape of the AMS and non-uniqueness of the controls apply to any control allocation method.

The representation of the AMS in spherical coordinates makes it possible to rapidly perform the on-line computations required by the direct allocation method. No prior information is required about the approximate location of the correct facet. Two options were discussed which have their respective advantages. The first option (facet box method) did not require large memory storage, but had a larger and variable number of on-line computations. The number of computations for a given control cycle will not exceed $n(n-1)$ box check comparisons (trivial) and a few 3D vector-matrix multiplications. The number of these matrix operations is uncertain but was found to not exceed 5 in our tests involving an aircraft model with 16 actuators.

The second option (facet table look-up method) required virtually no on-line computations and provided a guaranteed solution in a fixed time. The drawback was a potentially large memory requirement and longer off-line execution time. Overall, both options provide a considerable improvement over a sequential search of the facets.

Slight modifications to the spherical coordinate methods were shown to provide direct allocation solutions to coplanar systems. The properties of those methods were similar to those for non-coplanar systems. Overlapping sub-facets were identified by adding a third dimension to the table that stores the identifier of each sub-facet.

The advantage of the method over ganging and other simple control allocation techniques is that it guarantees the use of the maximum control authority available, while requiring very few on-line computations. Compared to other rapid search techniques for direct allocation, its advantage is a high degree of predictability and reliability. It is also ideal for lengthy simulations because of the extremely fast execution time. However, the method requires a significant amount of memory and is not well suited to reconfigurable control, which would require continuous update of the look-up table in real-time.

6. References

[1] J. Brinker & K. Wise, "Reconfigurable Flight Control for a Tailless Advanced Fighter Aircraft," *Proc. of the AIAA Guidance, Navigation, and Control Conference,* Paper AIAA 98-4107, Boston, MA, August 1998.

[2] R. Eberhardt & D. Ward, "Indirect Adaptive Flight Control of a Tailless Fighter Aircraft," *Proc. of the AIAA Guidance, Navigation, and Control Conference,* Paper AIAA 99-4042, Portland, OR, August 1999.

[3] D. Enns, "Control Allocation Approaches," *Proc. of the AIAA Guidance, Navigation, and Control Conference,* Paper AIAA 98-4109, Boston, MA, August 1998.

[4] W. C. Durham, "Constrained Control Allocation," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 4, 1993, pp. 717-725.

[5] W. C. Durham, "Constrained Control Allocation: Three-Moment Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 2, 1994, pp. 330-336.

[6] W. C. Durham, "Attainable Moments for the Constrained Control Allocation Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 17, No. 6, 1994, pp. 1371-1373.

[7] K. A. Bordignon & W. C. Durham, "Closed-Form Solutions to Constrained Control Allocation Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 5, 1995, pp. 1000-1007.

[8] J. Buffington & P. Chandler & M. Pachter, "Integration of On-line System Identification and Optimization-based Control Allocation," *Proc. of the AIAA Guidance, Navigation, and Control Conference,* Paper AIAA 98-4487, Boston, MA, August 1998.

[9] M. Bodson & W. Pohlchuck, "Command Limiting in Reconfigurable Flight Control," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 4, July-August 1998.

[10] W. C. Durham, "Computationally Efficient Control Allocation," *Proc. of the AIAA Guidance, Navigation, and Control Conference,* Paper AIAA 99-4214, Portland, OR, August 1999.

[11] J. Buffington, "Tailless Aircraft Control Allocation," *Proc. of the AIAA Guidance, Navigation, and Control Conference,* Paper AIAA 97-3605, New Orleans, LA, August 1997.

[12] F. P. Preparata & M. I. Shamos, *Computational Geometry, An Introduction,* Springer-Verlag, New York, 1985.

AIAA-01-xxxx

EVALUATION OF OPTIMIZATION METHODS FOR CONTROL ALLOCATION

Marc Bodson[*]
University of Utah, Electrical Engineering
50 S Central Campus Dr Rm 3280
Salt Lake City UT 84112
(801) 581 8590 bodson@ee.utah.edu

Abstract

The paper evaluates the performance and computational requirements of optimization methods for control allocation. Two control allocation problems are formulated: a direct allocation method that preserves the directionality of the moment, and a mixed optimization method that minimizes the error between the desired and the achieved moments as well as the control effort. The constrained optimization problems are transformed into linear programs, so that they may be solved using well-tried linear programming techniques such as the simplex algorithm. The paper discusses a variety of tricks that may be applied for the solution of the control allocation problem in order to accelerate computations. Performance and computational requirements are evaluated using aircraft models with different number of actuators, and with different properties. In addition to the two optimization methods, three algorithms with low computational requirements are also implemented for comparison: a redistributed pseudo-inverse technique, a quadratic programming algorithm, and a fixed-point method. The major conclusion of the paper is that constrained optimization may be performed with computational requirements that fall within an order of magnitude of those of simpler methods. The performance gains are small on the average, but sometimes significant. A variety of issues are discussed in the paper that affect the implementation of the various algorithms in a flight control system.

1. Introduction

In many flight control systems, ganging has been used to associate the three rotational degrees of freedom of an aircraft to its control surfaces. So, a pitching command is transformed into identical left and right elevator deflections, while a rolling command is transformed into opposite left and right aileron deflections. As advanced aircraft with many unconventional surfaces are being designed, the appropriate manner in which ganging should be implemented is becoming less obvious. Further, there is increased interest in control methods that are capable of reconfiguration after failures. In such cases, it is unlikely that a ganging method designed for an unfailed aircraft would be optimal after a failure. Therefore, there is a need for control allocation algorithms that perform *automatic* distribution of the control requirements among a large number of control surfaces, and exploit *all* the maneuvering capabilities of an aircraft given position and rate limits on the actuators.

[*] Marc Bodson is a Professor of Electrical Engineering and a Senior Member, AIAA. Effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-98-1-0013. The views and conclusions contained in the paper are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

The simplest control allocation methods are based on the unconstrained least-squares algorithm, and modifications of the solution aimed at accounting for position and rate limits [15], [23]. More complex methods formulate control allocation as a constrained optimization problem [5], [10]. Until recently, it was believed that optimization methods would be too complex, or too time-consuming, to be implemented in a flight control system [13], [15]. However, dramatic increases in computing speed, as well as more efficient algorithms, are rapidly changing that picture.

The objective of this paper is to evaluate optimization-based algorithms for control allocation, and to compare the results to those of simpler methods. The two main questions are: (1) whether these methods provide significant improvements of performance, and (2) whether the computational requirements place them out of reach. The optimization algorithms evaluated in the paper are well-tried methods of linear programming. They are implemented to solve both the direct allocation problem posed by Durham [10], and a more common error minimization objective [4]. A numerical evaluation is performed using a transport aircraft model and a tailless aircraft model. The transport aircraft model is evaluated with and without a ganging matrix that brings the number of actuators from 16 to 8, allowing to evaluate the cost and benefits of a higher number of actuators. The tailless model is of particular interest because it does not satisfy a condition that guarantees uniqueness of the solution in the direct allocation problem.

2. Control Allocation Problem

2.1 Control Allocation for Flight Control

We begin the paper by briefly discussing the motivation for control allocation. Consider the state-space model

$$\begin{aligned} \dot{x} &= Ax + Bu + d \\ y &= Cx \end{aligned} \tag{1.1}$$

where x, d, u, and y are all vectors. For the control of aircraft, the state vector x may include the angle of attack, the pitch rate, the angle of sideslip, the roll rate, and the yaw rate. The output vector y may contain the pitch rate, the roll rate, and the yaw rate. The control input vector u consists in the commanded actuator positions, or in the control surface deflections if the actuator dynamics are neglected. If the control variables are ganged, p=dim(u) may be as small as 3. Otherwise, the typical range is $p=5 \rightarrow 20$.

Model reference control laws, sometimes referred to as dynamic inversion control laws, rely on a reference model which represents the desired dynamics of the closed-loop system, for example

$$\dot{y}_M = A_M y_M + B_M r_M \tag{1.2}$$

where r_M is a reference input vector (determined by the pilot commands), and y_M represents the desired output of the system. Since the derivative of y is given by

$$\dot{y} = CAx + CBu + Cd \tag{1.3}$$

the objective may be achieved by setting

$$u = (CB)^{-1}\left(-CAx - Cd + A_M y + B_M r_M\right) \tag{1.4}$$

Model matching follows if the matrix CB is square and invertible, and if the original system is minimum phase. Adaptive implementations of this control law were discussed in the context of reconfigurable flight control in [1], [2].

If the matrix CB is not full rank, model matching may still be possible, but with a different model and a more complex control law. On the other hand, if CB is not square but full row rank (more columns than rows, as in the case of redundant actuators), the same model reference control law can be used if one defines

$$a_d = -CAx - Cd + A_M y + B_M r_M \tag{1.5}$$

and a control input u such that

$$(CB)u = a_d \tag{1.6}$$

Obtaining u from (1.6) requires that one solve a system of linear equations with more unknowns than equations. This may seem like an easy problem, but the difficulty in practice is that the vector u is constrained. The limits generally have the form

$$u_{\min,i} \le u_i \le u_{\max,i} \qquad \text{for } i = 1,\ldots,p \tag{1.7}$$

or, $u_{\min} \le u \le u_{\max}$, in vector form. Constraints originate from position and/or rate limits of the actuators. Given the limits, an exact solution may not exist, despite the redundancy. Further, whether an exact solution exists or not, the solution can generally *not* be assumed to be unique. We refer to the problem of finding a vector u that is the "best" possible solution of (1.6) within the constraints (1.7) as the *control allocation problem.*

The control allocation problem arises in other situations than the design of model reference control laws, specifically in the context of control laws designed using the concept of pseudo-effectors [23] and reconfigurable control in general [3], [13]. To discuss one approach to reconfigurable control, assume that a nominal control law has been designed so that a control input u_{nom} is produced under the assumption of a nominal matrix CB_{nom}. If a failure occurs, a new control input must be found that accounts for the modified matrix *CB*. In order to preserve the performance of the nominal control law, it is reasonable to set as an objective that

$$(CB)u = (CB)_{nom}\, u_{nom} \tag{1.8}$$

This problem fits into the previous format if we define the vector a_d such that

$$a_d = (CB)_{nom}\, u_{nom} \tag{1.9}$$

2.2 Mathematical Formulations of Control Allocation

In light of this preliminary discussion, we propose four mathematical formulations of the control allocation problem. These formulations take into account the fact that the exact solution of the control allocation problem may not exist, and that the solution may not be unique.

Direct Allocation Problem: given a matrix *CB*, find a real number ρ and a vector u_1 such that *J*=ρ is maximized, subject to

$$(CB)u_1 = \rho a_d \tag{1.10}$$

and $u_{\min} \le u \le u_{\max}$. If ρ>1, let $u = u_1/\rho$. Otherwise, let $u = u_1$.

Error Minimization Problem: given a matrix *CB*, find a vector u such that

$$J = \|CBu - a_d\| \tag{1.11}$$

is minimized, subject to $u_{\min} \le u \le u_{\max}$.

Control Minimization Problem: given a matrix *CB* and a vector u_p, and given a vector u_1 such that $u_{\min} \le u_1 \le u_{\max}$, find a vector u such that

$$J = \|u - u_p\| \tag{1.12}$$

is minimized, subject to

$$(CB)u = (CB)u_1 \tag{1.13}$$

and $u_{\min} \le u \le u_{\max}$.

Mixed Optimization Problem: given a matrix *CB* and a vector u_p, find a vector *u* such that

$$J = \|CBu - a_d\| + \varepsilon \|u - u_p\| \tag{1.14}$$

is minimized, subject to $u_{\min} \le u \le u_{\max}$.

2.3 Discussion of the Direct Allocation Problem

The *direct allocation problem* was proposed by Durham [10]. The objective is to find a control vector *u* that results in the best approximation of the vector a_d, *in the given direction*. The implicit assumption is that directionality is an important characteristic of multivariable control systems, and of flight control in particular.

Numerical algorithms for direct allocation have been proposed. The original algorithm [10] was slow and difficult to implement, but an elegant approach [11] reduced the number of computations considerably. A fast implementation using spherical coordinates and look-up tables was proposed in [21], although it requires a good number of off-line computations. Another fast technique was proposed in [12]. Unfortunately, the algorithm is hard to reproduce, based on the limited information available in the paper, and it is not guaranteed to converge to the solution. Linear programming algorithms discussed later in this paper avoid most of these problems.

An interesting feature of the direct allocation problem is that its solution is unique under a relatively mild condition on the matrix *CB*. Specifically, the condition is:

Any q columns of the CB matrix are linearly independent (linear independence condition)

where *q* is the number of rows of *CB*. Durham considered the problem where $q=3$, which is typical of flight control. In that case, the *3* components of a_d in the model reference control law with *q*, *p*, and *r* as outputs are three desired rotational accelerations (Durham refers to them as moments, which corresponds to a slightly different formulation). The columns of the *CB* matrix then represent the contributions of the individual actuators towards these desired accelerations. An interpretation of the linear independence condition is that *no three actuators produce coplanar acceleration vectors*.

In many applications, the linear independence condition is satisfied. However, there are cases where it is not. In general, the accelerations produced by pitch thrust vectoring and by two symmetric pitch surfaces (such as elevators) are coplanar. Similarly, a failed actuator yields a zero column in the *B* matrix, which automatically violates the linear independence condition. It has been argued that the *CB* matrix could be perturbed by small numbers to enforce the condition, but this fix is likely to induce numerical sensitivity. However, the direct allocation concept can be extended to systems that do not satisfy the linear independence condition, and an extension of the method to the coplanar case is discussed in [22].

Another difficulty of the direct allocation formulation is that it requires

$$u_{\min} \le 0 \qquad \text{and} \qquad u_{\max} \ge 0 \tag{1.15}$$

(to be interpreted as vector inequalities), which makes application difficult in the case of rate-limited actuators. A solution to this problem consists in applying the technique to increments of the vector *u*. However, this solution introduces a wind-up problem, which must be resolved using "restoring" techniques. The direct allocation method has also been criticized for not enabling axis prioritization. However, it presents interesting advantages, including a guarantee of maximum control utilization combined with the existence of fast computational procedures.

2.4 Discussion of the Error Minimization, Control Minimization, and Mixed Optimization Problems

The *error minimization problem* is the most commonly encountered formulation of control allocation. The l_2 norm or Euclidean norm is typically used in (1.11), although the l_1 norm has also been proposed in order to use linear programming techniques. Often, a weighting matrix is inserted in the norm to prioritize the axes.

The *control minimization problem* is a secondary optimization objective to be satisfied if the solution of the primary objective is not unique. The vector u_p represents some preferred position of the actuators (*e.g.*, zero deflections). After a solution yielding minimum error is obtained, the solution of minimum deviation from the preferred position is picked among all equivalent solutions. A weighting matrix may also be incorporated in the norm, to prioritize the actuators.

The control minimization objective has been considered as part of a multi-branch optimization algorithm [5]. Specifically, the secondary optimization problem was solved (only) when the solution of the primary error minimization problem was *J=0*. Indeed, the solution is typically not unique in that case. One should note, however, that if the linear independence condition is not satisfied, the solution of the primary objective is generally not unique, regardless of the feasibility of the desired vector a_d. Therefore, it is reasonable to perform control minimization no matter what the result of the primary error minimization step is.

The *mixed optimization problem* combines the error and control minimization problems into a single problem, through the use of a small parameter ε. If the parameter ε is small, priority will be given to error minimization over control minimization, as desired. Often, the combined problem may be solved faster than the error and control minimization problems solved sequentially, and with better numerical properties.

3. Three Simple Algorithms for Control Allocation

The formulations of control allocation represent them as constrained nonlinear optimization problems. Because of the limited number of variables and the convexity of the constraint set, the optimization problems are simple from a modern computational perspective. However, computations must be performed at high rate (around 100 Hz) and in the midst of many other control computations. Further, predictable reliability is required for safety-critical systems, and human intervention is not possible. For those reasons, the most common implementations of control allocation have relied on approximations of the error minimization objective. We discuss three approaches that have been proposed.

3.1 Redistributed Pseudo-Inverse

The *redistributed pseudo-inverse method* of [13], [23] begins by finding the control vector *u* that minimizes

$$J = \|u\|_2^2 \tag{1.16}$$

subject to

$$(CB)u = a_d \tag{1.17}$$

This optimal vector is given through the pseudo-inverse of the *CB* matrix, with

$$u = (CB)^T \left[(CB)(CB)^T\right]^{-1} a_d \tag{1.18}$$

If the control vector satisfies the constraints, the algorithm stops. Otherwise, the components of the control vector that exceed the limits are clipped at their allowable values, and the inverse is re-computed with the components that did not reach their limits. The procedure is repeated until all components have reached their limits, or until the solution of the reduced least-squares problem satisfies the constraints. Computations may be performed by modifying a_d with the contributions of the saturated controls, and zeroing the columns of the CB matrix associated with these controls. Because the modified matrix $(CB)(CB)^T$ will typically become singular at some point, it is necessary to replace it by a regularized matrix $(CB)(CB)^T + \varepsilon I$.

The redistributed pseudo-inverse method is very simple, and very effective. However, it does not guarantee full utilization of the actuators' capabilities. For example, consider the desired vector and parameters

$$a_d = \begin{pmatrix} 0 \\ 9 \\ 0 \end{pmatrix}, \quad CB = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{1.19}$$

with the actuator limits $u_{\max} = -u_{\min} = \begin{pmatrix} 5 & 10 & 2 & 1 \end{pmatrix}$. The solution of the unconstrained least-squares is $u_{LS}^T = \begin{pmatrix} 0 & 6 & -3 & 3 \end{pmatrix}$, which gives the clipped solution $u^T = \begin{pmatrix} 0 & 6 & -2 & 1 \end{pmatrix}$. At the second iteration, the control vector $u^T = \begin{pmatrix} 0 & 8 & -2 & 1 \end{pmatrix}$ is obtained. The acceleration vector $a^T = \begin{pmatrix} 0 & 9 & -1 \end{pmatrix}$ is achieved, instead of the desired vector. However, the desired vector a_d may be attained using only the second control variable, specifically

$$u^T = \begin{pmatrix} 0 & 9 & 0 & 0 \end{pmatrix} \Rightarrow a^T = \begin{pmatrix} 0 & 9 & 0 \end{pmatrix} \tag{1.20}$$

What this example shows is that some bad choices may be made early in the iterations of the method, which cannot be recovered from, and prevent full utilization of the control authority.

3.2 Quadratic Programming

Quadratic programming generally refers to the numerical solution of the optimization problems with an l_2 norm. Enns [15] proposed several approaches, including approximate methods with low numerical requirements. We consider one of the simplest algorithms here, which begins by finding the control vector u that minimizes

$$J = \|CBu - a_d\|_2^2 \tag{1.21}$$

The solution is again given by the pseudo-inverse, and if the vector satisfies the constraints, the algorithm stops. Otherwise, the method continues in a manner different from the redistributed pseudo-inverse. The optimal solution subject to the equality constraints

$$\|u\|_2^2 = p \tag{1.22}$$

is computed, where p is the dimension of u. This step assumes that an affine transformation has been applied to the data of the problem in order to replace the constraint $u_{\min} \le u \le u_{\max}$ by $\|u\|_\infty = 1$. Then, the l_2 constraint $\|u\|_2^2 = p$ is applied as an approximation of the box constraint $\|u\|_\infty = 1$ (the l_2 constraint is such that the sphere contains the box while touching its corners).

The solution to the problem with the equality constraint is

$$u = (CB)^T \left[(CB)(CB)^T + \lambda I \right]^{-1} a_d \tag{1.23}$$

which is reminiscent of the regularized pseudo-inverse. However, λ is not a small constant, but a Lagrange multiplier that must be found by solving the equation

$$\gamma(\lambda) = \left\| (CB)^T \left[(CB)(CB)^T + \lambda I \right]^{-1} a_d \right\|_2^2 = 1 \tag{1.24}$$

It turns out [15] that the function γ(λ) is monotonically decreasing, and that the solution λ may be found using only a few iterations of a bisection method. The control vector u is then clipped to satisfy the constraints.

The advantage of the method is its numerical simplicity. However, the replacement of the l_∞ constraint by an l_2 constraint implies that the solution is not exact. Clipping must be applied to enforce the actual constraint, and may yield poor results. Also, the origin of the transformed set corresponds to a non-zero vector whenever some surfaces have non-symmetric limits (in particular, spoilers). As a result, the control surface deviations produced by the algorithm are usually off-set towards undesirable values. While this problem can be addressed by the use of additional steps in the algorithm (secondary optimization), the algorithm then becomes more complex and less attractive.

3.3 Fixed-Point Method

The *fixed-point method* of [7] finds the control vector u that minimizes

$$J = (1-\varepsilon) \left\| (CB)u - a_d \right\|_2^2 + \varepsilon \left\| u \right\|_2^2 \tag{1.25}$$

subject to $u_{\min} \le u \le u_{\max}$. This problem is a special case of the mixed optimization with an l_2 norm and u_p=0. The algorithm proceeds by iterating on the equation

$$u_{k+1} = sat\left[(1-\varepsilon)\eta(CB)^T a_d - (\eta M - I)u_k \right] \tag{1.26}$$

where

$$M = (1-\varepsilon)(CB)^T(CB) + \varepsilon I, \quad \eta = \frac{1}{\|M\|_2} \tag{1.27}$$

and *sat* is the saturation function that clips the components of the vector u to their allowable values.

The fixed-point algorithm is extremely simple, and much of the computations need to be performed only once, before iterations start. Remarkably, the algorithm also provides an *exact* solution to the optimization problem, and it is guaranteed to converge. Its drawback is that convergence of the algorithm may be very slow and strongly depends on the problem (the number of iterations required may vary by orders of magnitude depending on the desired vector). In addition, the choice of the parameter ε is delicate, as it affects the trade-off between the primary and the secondary optimization objectives, as well as the convergence of the algorithm.

4. Formulation of Control Allocation Problems as Linear Programs

4.1 Linear Programs

Buffington [5] showed that the error minimization and the control minimization problems could be reformulated as linear programs, assuming that the norm used was the l_1 norm. The problems could then be solved *exactly*, using standard linear programming software. A standard linear programming (LP) problem consists in finding a vector x such that

$$J = c^T x \tag{1.28}$$

is minimized, subject to

$$0 \le x \le h, \quad \text{and} \quad Ax = b \tag{1.29}$$

In this equation, vector inequalities are to be interpreted element-by-element. Alternative formulations exist, replacing $0 \le x \le h$ by $x \ge 0$, and $Ax = b$ by $Ax \ge b$. However, these differences are not significant and the present form will be adequate for our discussion. Note that the matrix A and the vector x are **not** the same as the state-space variables defined earlier. This slight abuse of notation will enable us to use of standard LP notation.

4.2 Conversion of the Direct Allocation Problem to an LP Problem

The conversion of the control allocation problems to LP problems is not immediately obvious, and several formulations may be derived. It is generally desirable to obtain a formulation with as few rows in the matrix A as possible, and we present an approach that requires the smallest number of rows. Direct allocation specifies q linear equations

$$a_{d,i} = \rho CB_i u, \quad i = 1, \ldots, q \tag{1.30}$$

where CB_i is the ith row of CB, and $a_{d,i}$ is the ith element of a_d. Re-ordering the rows of CB and of a_d so that the first element of a_d is the one with the largest magnitude, and eliminating the variable ρ, gives a new system of q-1 linear equations

$$\left(a_{d,i} CB_1 - a_{d,1} CB_i\right) u = 0, \quad i = 2, \ldots q \tag{1.31}$$

which may be written in matrix form as

$$M \cdot CB \cdot u = 0, \quad \text{with } M = \begin{pmatrix} a_{d,2} & -a_{d,1} & 0 & \cdots & 0 \\ a_{d,3} & 0 & -a_{d,1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ a_{d,q} & 0 & 0 & \cdots & -a_{d,1} \end{pmatrix} \tag{1.32}$$

These equality constraints are equivalent to the original conditions if $a_{d,1} \ne 0$. Unless $a_d = 0$, a trivial case requiring u=0, this situation will not be encountered.

Define $x = u - u_{\min}$, so that $0 \le x \le u_{\max} - u_{\min}$ and notice that the optimization criterion may be written as

$$\max\left(\rho = \frac{\|CBu\|}{\|a_d\|}\right) = \min\left(J = -a_d^T CBu\right) \tag{1.33}$$

given that CBu is proportional to a_d. Then, the LP problem may be defined by the following matrices

$$\begin{aligned} A &= M \cdot CB, \quad b = -Au_{\min} \\ c^T &= -a_d^T CB, \quad h = u_{\max} - u_{\min} \end{aligned} \tag{1.34}$$

If a solution is found to the LP problem, the solution to the direct allocation problem may be obtained using

$$u = x + u_{\min}, \quad \rho = a_d^T CBu / \|a_d\|^2$$
$$u = u/\rho \ \text{ if } \rho > 1 \tag{1.35}$$

Note that the direct allocation problem may be solved as an LP problem for an arbitrary number of rows in the matrix *CB*, as opposed to the algorithms proposed earlier. In particular, direct allocation could be performed for forces as well as moments in a flight control application that demanded it. In the standard direct allocation problem where the number of rows of *CB* is 3, the number of rows of the matrix *A* is only 2, which makes the size of the LP problem very small. Linear programming theory implies that optimal solutions to the problem will be such that all variables except 2 will be at their limits (before the final division by ρ). This observation was made earlier by Durham using different arguments. Linear programming theory also indicates that even if the linear independence assumption is not satisfied, at least one of the optimal solutions will be such that all variables except 2 are at their limits.

4.3 Conversion of the Mixed Optimization Problem to an LP Problem

The error minimization and control minimization problems can also be converted to LP problems, as shown in [5]. This conversion requires that the norm used in the optimization criterion is the l_1 norm. Interestingly, various LP problems may be obtained for the same control allocation problem, and formulations smaller than those of [5] may be obtained. Also, from multiple investigations, we concluded that the mixed optimization problem yielded the most useful formulation, and one that could be solved most effectively. Therefore, we only discuss this problem.

First, we define the function $s(x)$

$$\begin{aligned} s(x) &= x \quad \text{if } x > 0 \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{1.36}$$

This function is to be interpreted element-by-element in the vector case. We also assume that the preferred vector satisfies $u_{\min} \le u_p \le u_{\max}$. This condition may be eliminated easily, once the technique is understood. Define

$$u^+ = s(u - u_p), \quad u^- = -s(u_p - u) \tag{1.37}$$

so that

$$u = u^+ - u^- + u_p, \quad 0 \le u^+ \le u_{\max} - u_p, \quad 0 \le u^- \le u_p - u_{\min} \tag{1.38}$$

Similarly, define

$$e = CBu - a_d, \quad e^+ = s(e), \quad e^- = -s(-e) \tag{1.39}$$

so that

$$e = e^+ - e^-, \quad 0 \le e^+ \le e_{\max}, \quad 0 \le e^- \le e_{\max} \tag{1.40}$$

where $e_{\max}$ is some upper bound on the achievable error, *e.g.*, $e_{\max} = \|CBu_p - a_d\|_1$.

With these definitions, the optimization problem involves a system of n linear equations

$$e^+ - e^- - CBu^+ + CBu^- = CBu_p - a_d \tag{1.41}$$

and the cost criterion

$$J = \sum_{i=1}^{q} e_i^+ + \sum_{i=1}^{q} e_i^- + \varepsilon \sum_{i=1}^{p} u_i^+ + \varepsilon \sum_{i=1}^{p} u_i^- \tag{1.42}$$

Therefore, defining the vector $x^T = \begin{pmatrix} e^+ & e^- & u^+ & u^- \end{pmatrix}$, the LP problem is specified by

$$\begin{aligned} &A = \begin{pmatrix} I & -I & -CB & CB \end{pmatrix}, \quad b = CBu_p - a_d \\ &c^T = \begin{pmatrix} 1 & \dots & 1 & \varepsilon & \dots & \varepsilon \end{pmatrix} \\ &h^T = \begin{pmatrix} e_{\max} & e_{\max} & u_{\max} - u_p & u_p - u_{\min} \end{pmatrix} \end{aligned} \tag{1.43}$$

Note that the A matrix of the LP problem has as many rows as the CB matrix (one more row than for direct allocation). Arbitrary problems can be solved, but for the standard 3-moment case, the dimension is only 3. The problem is therefore very manageable, and only slightly more complex than the direct allocation problem. As for direct allocation, linear programming theory implies certain properties of the solution. All the components of the optimal vector x except 3 will be at either the upper limit or the lower limit. In terms of the control vector, this property implies that all but three control variables will be either at the upper limit, or at the lower limit, or at the preferred position. If the vector a_d cannot be achieved in any direction, all the control variables will be at one of the limits or at the preferred positions. The desirability of the property may be debated: on the one hand , it is desirable to see the algorithm use only effective surfaces [23], and on the other hand, it is desirable to see all surfaces move together to achieve the desired moment [8]. It has been argued that this property was detrimental to on-line identification, since some surfaces may not be used at all. However, we will see in the numerical section of this paper that the conclusion is only valid if the commanded accelerations are small.

5. Implementation using the Simplex Algorithm

5.1 Simplex Algorithm

A well-established method for solving LP problems is the *simplex algorithm* [18]. The algorithm is guaranteed to find an optimal solution in a finite period of time, it is easy to code, and it works well in practice. In the case of the direct allocation problem, the simplex algorithm applied to the associated LP problem constitutes an alternative to the implementation of [12], which guarantees a solution in a finite period of time. The simplex algorithm can also be applied to the mixed optimization problem. Speed of execution can be maximized by taking advantage of the particular aspects of the LP problem.

An important definition in the context of the simplex algorithm is that of a basic feasible solution. A vector x is called a *feasible solution* of the LP problem if $0 \le x \le h$ and $Ax = b$. A vector x is called a *basic feasible solution* if it is a feasible solution and $n - m$ of its variables at their upper or lower limits, where m is the number of rows of A and n is the number of columns of x (n is also the number of elements of x). It can be shown that if there is a unique optimal solution, it is a basic feasible solution. Further, if there exists a set of optimal solutions, at least one of the solutions is a basic feasible solution.

The idea of the simplex algorithm is to start from a basic feasible solution and, at every step, to find a new basic feasible solution with a lower cost. From one step to the next, only one of the variables that is at a limit is swapped with one that is not. When it is not possible to decrease the cost, the algorithm stops. It can be proved that the resulting solution is optimal. Because the possible number of elements in the set of basic feasible solutions is finite, the number of iterations is bounded by a known quantity. Specifically, the number of iterations is at most $n!/(m!(n\text{-}m)!$. For a direct allocation problem in a 3-dimensional space with p control variables, $n=p$ and $m=2$. For a mixed optimization problem in a 3-dimensional space with p control variables, $n=2p+6$ and $m=3$. Practically, however, convergence occurs well before the maximum number of iterations is reached.

Various implementations of the simplex algorithm are possible. However, it may be noted that the control allocation problems under consideration are such that $n \gg m$. Therefore, the problems are well suited for the so-called *revised simplex method* ([18], p. 60). The number of computations in that method depend only moderately on the number of columns m. Also, most variables of the vector x naturally have both upper and lower bounds (which is the reason why we defined the LP problem with both bounds, although the problem with lower bounds only is more commonly encountered). If an LP code only handles lower bounds of the type $x \geq 0$, a constraint $0 \leq x_1 \leq h$ can be handled by adding a so-called *slack variable,* with the constraints $x_1 \geq 0$, $x_2 \geq 0$, and $x_1 + x_2 = h$. However, this approach increases the number of equality constraints, and by a particularly large number for the problems under consideration. For direct allocation, *m* would become *p+2* instead of *2*, and *n* would become *2p* instead of *p*. The variable *p* is the number of actuators, and therefore $p \gg 2$. Instead of using slack variables, an implementation of the algorithm with upper and lower bounds avoids this problem ([18], p. 53). It appears that the revised simplex algorithm with upper and lower bounds may be close to the one used in [16], although too few details are available in the paper to be sure.

5.2 Refinements to the Algorithm

Several tricks can be used to accelerate execution. First of all, note that the algorithm must be initialized with a basic feasible solution. Typically, this is achieved during an initialization phase which, itself, requires the use of the simplex algorithm. A vector x_S of slack variables is introduced so that $Ax + x_s = b$, and the algorithm is used to eliminate x_S by setting $J = \|x_S\|$. For the direct allocation problem, this step requires the preliminary solution of an LP problem with $n=p+2$ and $m=2$ variables.

Interestingly, the algorithm for mixed optimization can be initialized directly, by-passing the need for a two-step procedure. Specifically, the algorithm may be initialized with $u=u_p$, so that

$$\begin{aligned} u^+ &= 0, \quad e^+ = s(CBu_p - a_d) \\ u^- &= 0, \quad e^- = -s(-CBu_p + a_d) \end{aligned} \tag{1.44}$$

Note that the resulting vector *x* has all but *n* of its elements at zero (at least), and solves *Ax=b*. Therefore, it is a basic feasible solution. Because of this special feature, we have found that the mixed optimization problem could be solved in approximately the same time as the direct allocation problem.

Another useful trick to speed-up computations is the Sherman-Morrison-Woodbury formula ([19], p. 605). The revised simplex method requires the inversion of the square matrix obtained by taking the columns of the *A* matrix that correspond to the non-basic variables (the variables that are not at the limits). However, only one column of the matrix changes from one iteration to the next. Therefore, the inverse of the matrix can be computed recursively using the inverse at the previous iteration, requiring much fewer computations than the full matrix inverse. This algorithm is particularly attractive if the matrix has a known inverse at the initial step. Fortunately, modulo some sign changes, the initial matrix is simply the identity matrix for the mixed optimization problem.

Another issue to be dealt with is the fact that the finite-time convergence of the simplex algorithm is based on the assumption that the cost is monotonically decreasing. If such is not the case, the algorithm may cycle among a set of basic feasible solutions having identical costs. Cycling is a singular condition that requires two real variables to be exactly equal. Although one would expect this situation to be extremely rare, it was rapidly encountered in our experiments of control allocation. The reason was that the aircraft models had identical control derivatives for symmetric surfaces, and that a large number of cases were computed in a small period of time. Fortunately, anticycling procedures have been studied that help to deal with this problem (see, in particular, [18] p. 78 and 83, and [17], p. 446). Some work, however, is required to adapt the methods to the revised simplex method with upper and lower bounds.

6. Numerical Results

6.1 General Conditions of the Numerical Evaluation

The methods of control allocation were evaluated numerically on several examples. Three aircraft models were used. The first model is a C-17 model with 8 actuators, and four of the actuators represent ganged surfaces. The control effectors are left elevators, right elevators, a left aileron. a right aileron, a lower rudder, an upper rudder, left spoilers, and right spoilers. The second model is the same C-17 model, but with no ganging, resulting in 2 left elevators, 2 right elevators, a left aileron, a right aileron, a lower rudder, an upper rudder, 4 left spoilers, and 4 right spoilers. The third model is the model of a tailless aircraft found in [2]. It is based on Lockheed's ICE (*innovative control effectors*) model, and has 11 actuators: left elevon, right elevon, pitch flaps, left all-moving tip, right all-moving tip, pitch thrust vectoring, yaw thrust vectoring, left spoiler slots, right spoiler slots, left outboard leading-edge flaps, and right outboard leading-edge flaps.

The C-17 model satisfies the linear independence condition (*i.e.,* it does not have coplanar controls). The tailless model does not: pitch thrust vectoring and pitch flaps produce pure pitching accelerations, and therefore linearly dependent vectors. Each of these control variables is also linearly dependent on the left and right elevons. As a result, the control variables needed to produce an acceleration vector are not unique, whether the vector is feasible or not. Given a control allocation solution, for example, it is easy to obtain a different one where the effect of a deflection of pitch flaps is canceled by pitch thrust vectoring.

Although rate limits may easily be incorporated in the mixed l_1 optimization method, the analysis of this paper only considers position limits, in order to make the comparisons more tractable and meaningful. The results assess the ability of the algorithms to reach the whole set of attainable accelerations. The interactions between the algorithms and a closed-loop flight control system are also not addressed (see [10] for some results in that direction).

The parameters used in the algorithms were as follows. In the redistributed pseudo-inverse, $\varepsilon = 10^{-4}$. In the quadratic programming method, a tolerance of 10^{-6} was used for the stopping rule of the bisection method in the γ equation. For the fixed-point method, the number of iterations was set at 50, with $\varepsilon = 10^{-3}$. Note that the ε parameter must be large enough to get good convergence properties, and small enough that the control minimization objective remains secondary to the error minimization objective. In the mixed optimization objective, ε does not have an impact on convergence, although it should not be too small with respect to the numerical precision of the machine. A value of 10^{-6} was chosen.

The results were obtained on a Pentium III machine running at 500 MHz. Most results were obtained using implementations of the algorithms as m-files in Matlab 5.3. Timing results were obtained using the tic/toc commands, but are given for comparison purposes only. To obtain timing information for the simplex algorithm, the mixed optimization was coded in Fortran, yielding much faster execution times.

6.2 C-17 Model

Table 1 shows the results for the C-17 model with 8 actuators. The five methods discussed in the paper are listed: the redistributed pseudo-inverse technique, the quadratic programming algorithm, the fixed-point method, the direct allocation algorithm based on the simplex, and the mixed optimization algorithm using the simplex. The table gives the results obtained for 1000 randomly distributed vectors. The components of the vectors were evenly distributed in the range [±40, ±20, ±5] deg/s^2 (the vectors are associated to rotational accelerations in the pitch/roll/yaw axes). A procedure similar to the one described in [10] was used to plot the set of achievable accelerations, and the ranges were selected by visual inspection of the set, so that all the vectors in the set would be feasible.

The columns of Table 1 give the values of the average error $\|CBu - a_d\|_2$ over 1000 points, the maximum error $\|CBu - a_d\|_2$, the average execution time, and the average norm of the control $\|u\|_2$. Since u_p was set at zero, the last column is a measure of the secondary performance objective. Note that the norm that was used for evaluation was the l_2 norm, which favors the first three methods (the mixed optimization method minimizes the l_1 norm).

One may first notice that the average and maximum errors are zero for the direct allocation and mixed optimization methods, confirming the fact that the random vectors were indeed attainable, and that the optimization methods are able to find an exact solution, if one exists. The average errors for the other methods are small. Note that part of the error for the fixed-point method may be due to the trade-off between the error minimization and the control minimization. Interestingly, the mixed optimization method using the simplex (which is based on the l_1 norm) does *not* yield such a trade-off. In other words, although this algorithm mixes the two objectives, the final result is similar to the one that would be obtained by first solving an error minimization objective, and then a control minimization objective.

The column with the maximum errors shows a large maximum error for the redistributed pseudo-inverse. In other experiments, even larger errors were found. For example, the following pair was found

$$a_d = \begin{pmatrix} 23.7 & 12.7 & -4.9 \end{pmatrix} \Rightarrow a = \begin{pmatrix} 0.19 & 18.6 & -3.9 \end{pmatrix} \qquad (1.45)$$

yielding an error norm of 24.2 deg/s^2. Yet, the desired vector is exactly feasible! This example, as the one in section 3.1, is such that the algorithm makes a bad choice in one of the iterations, and never recovers. Analyzing the example, it was found that the control vector applied a differential elevator deflection, while realization of the pitching moment required full symmetric deflection. It was also found that such large errors occurred in only small subsets of the acceleration space. Indeed, the average error is very small. In practice, it is likely that such large but infrequent errors would not degrade significantly the closed-loop performance, but they could be the source of glitches in the control variables, such as those observed in [23].

The timing data should be interpreted with caution, since it was obtained with an interpreted language. However, it is useful for comparison of the methods. The timing information shows that the computational requirements of both optimization methods are within an order of magnitude of the simpler methods. Requirements for direct allocation and mixed optimization are also comparable. Although direct allocation involves a smaller LP problem, the mixed optimization by-passes the initialization step.

The last column of Table 1 shows the control deflections required to achieve the desired moments. Larger numbers are found for quadratic programming and for direct allocation. Regarding quadratic programming, the single-pass algorithm considered here does not minimize the norm of the control vector, and non-symmetric control surfaces such as spoilers off-set the solution towards a non-zero vector. A secondary optimization could be performed, but would take away from the simplicity of the method. Regarding direct allocation, the larger control deflection are an inherent feature of the method, which does not attempt to minimize the control deflections.

Table 2 is similar to Table 1, except that a larger set of desired accelerations is used, with values in the range [±80,±40,±10] deg/s^2. This set contains infeasible accelerations, so that all errors in the table are nonzero. The improvements provided by optimization are noticeable in the table (10% to 20% on the average). Recall that the norm used for comparison is the l_2 norm, so that the comparison favors the first three methods. Timing shows slightly larger numbers than for the feasible case, but overall, the trends are similar to the previous case.

Method	Average error (°/s^2)	Maximum error (°/s^2)	Average time (ms)	Average control (°)
Redistributed pseudo-inverse	0.24	14.5	3.13	16.4
Quadratic programming	0.05	2.49	2.31	30.1
Fixed-point method	0.16	2.68	5.71	15.0
Direct allocation (simplex)	0.00	0.00	15.2	28.8
Mixed l_1 optimization	0.00	0.00	15.3	17.7

Table 1: Control allocation results. C-17 aircraft model with 8 actuators. Feasible set.

Method	Average error (°/s^2)	Maximum error (°/s^2)	Average time (ms)	Average control (°)
Redistributed pseudo-inverse	10.1	48.0	4.12	41.7
Quadratic programming	10.5	39.5	4.50	40.2
Fixed-point method	8.97	34.6	6.37	34.0
Direct allocation (simplex)	9.00	41.4	17.0	48.0
Mixed l_1 optimization	7.79	35.7	20.2	44.7

Table 2: Control allocation results. C-17 aircraft model with 8 actuators. Infeasible set.

Table 3 and Table 4 are similar to Table 1 and Table 2, respectively. The C-17 model was replaced by the original model without ganging of the actuators. Although the number of actuators doubled, the timing information indicates a computational increase that is quite less than a factor of two. Other trends in the data are generally similar to the previous case. An interesting number is the average error provided by mixed optimization, which is 5.98 deg/s^2 instead of 7.78 deg/s^2 in the previous case (a reduction of almost 25%). This increase in performance is solely due to the "unganging" of the spoilers and elevators.

6.3 Tailless Model

Table 5 and Table 6 are similar to Table 1 and Table 2, respectively, but with the C-17 model replaced by the tailless model. The C matrix was the same as before, so that the vector a_d remained a pitch/roll/yaw acceleration vector. The ranges of desired vectors were [±150, ±200, ±15] deg/s^2 for the feasible case, and [±300,±400,±30] deg/s^2 for the infeasible case. The redistributed pseudo-inverse method does not exhibit the large errors that it did for the C-17 model, but this may be true only because the ranges for the feasible case were smaller in relation to the boundary of the feasible set. On the average, a 20% improvement or more is gained using mixed optimization.

Note that the tailless model is such that the linear independence condition is not satisfied. Although the original direct allocation method assumed such condition, the optimization objective may be defined without the condition. The geometry of the set of attainable vectors is different, and the solution is not unique on the boundary [22]. Nevertheless, the simplex algorithm is capable of finding (one of) the solutions in that case.

6.4 Realistic Timing Data

Table 7 addresses more realistically the issue of computational feasibility and, in particular, the two questions: (1) what is the computational requirement of the mixed optimization algorithm in a compiled language, and (2) how much larger is the worst-case requirement compared to the average requirement. Here, the C-17 model with 8 actuators and the tailless model with 11 actuators are used, with the feasible as well as the infeasible sets. Only the mixed optimization algorithm was implemented, because of its particular interest in light of the previous results. Evaluation was performed with 1000 random vectors. However, for this analysis, each vector was computed 1000 times to obtain reliable timing data.

Generally, the errors reported in Table 7 are comparable to the ones obtained earlier. They are not exactly identical because of some small differences in implementation and because of different random numbers. Also, the Fortran code used a single precision format. Single precision was used to demonstrate that the method was not exceedingly sensitive to small numerical errors. Indeed, while small errors are found in the feasible case, they are comparable to the machine precision. Double precision was also tried with very little increase in computing time, and resulted in negligible error numbers for the feasible sets.

Timing data shows execution times much smaller than in the interpreted Matlab environment. The times are well within the capabilities of existing computers and future flight control systems (confirming the conclusions of [16]). The worst-case timing requirement is 3 times the average value, a property which was verified to remain true for 10,000 random vectors. Trends in the timing requirements for feasible *vs.* infeasible sets and C-17 *vs.* tailless models are comparable to those observed in Matlab. One may have confidence that the methods would exhibit comparable performance with other aircraft models, and could be implemented easily with much greater number of actuators.

Method	Average error (°/s^2)	Maximum error (°/s^2)	Average time (ms)	Average control (°)
Redistributed pseudo-inverse	0.09	14.0	4.12	21.4
Quadratic programming	0.18	3.28	2.58	49.3
Fixed-point method	0.12	1.89	7.25	20.2
Direct allocation (simplex)	0.00	0.00	18.7	39.9
Mixed l_1 optimization	0.00	0.00	17.9	26.2

Table 3: Control allocation results. C-17 aircraft model with 16 actuators. Feasible set.

Method	Average error (°/s^2)	Maximum error (°/s^2)	Average time (ms)	Average control (°)
Redistributed pseudo-inverse	8.20	47.1	5.82	52.2
Quadratic programming	9.46	40.2	3.63	58.0
Fixed-point method	7.17	34.4	7.91	43.6
Direct allocation (simplex)	6.76	40.1	21.2	66.8
Mixed l_1 optimization	5.98	35.7	25.5	61.5

Table 4: Control allocation results. C-17 aircraft model with 16 actuators. Infeasible set.

Method	Average error (°/s^2)	Maximum error (°/s^2)	Average time (ms)	Average control (°)
Redistributed pseudo-inverse	0.00	0.02	3.79	27.7
Quadratic programming	2.07	65.5	2.47	61.0
Fixed-point method	3.26	9.19	6.70	25.8
Direct allocation (simplex)	0.00	0.00	18.1	43.1
Mixed l_1 optimization	0.00	0.00	17.4	31.9

Table 5: Control allocation results. Tailless aircraft model with 11 actuators. Feasible set.

Method	Average error (°/s^2)	Maximum error (°/s^2)	Average time (ms)	Average control (°)
Redistributed pseudo-inverse	15.3	197.	4.51	68.7
Quadratic programming	44.4	234.	2.91	79.0
Fixed-point method	17.4	158.	6.76	54.3
Direct allocation (simplex)	15.2	196.	18.2	81.4
Mixed l_1 optimization	11.5	180.	20.0	72.9

Table 6: Control allocation results. Tailless aircraft model with 11 actuators. Infeasible set.

Aircraft model	Average error (°/s^2)	Maximum error (°/s^2)	Average time (μs)	Maximum time (μs)
C-17 (8) Feasible set	$1.8\ 10^{-6}$	$3.9\ 10^{-4}$	83	172
C-17 (8) Infeasible set	7.6	36.1	121	281
Tailless Feasible set	$1.1\ 10^{-5}$	$2.4\ 10^{-4}$	90	219
Tailless Infeasible set	8.8	190	132	328

Table 7: Control allocation results with mixed l_1 optimization. C-17 aircraft model with 8 actuators and tailless model. Feasible and infeasible sets. FORTRAN implementation.

Method	CN(R_u) Small set	CN(R_u) Feasible set	CN(R_u) Infeasible set
Redistributed pseudo-inverse	$2.1\ 10^{16}$	$2.3\ 10^{3}$	194.
Quadratic programming	$1.5\ 10^{18}$	$3.3\ 10^{16}$	$2.9\ 10^{4}$
Fixed-point method	$3.9\ 10^{16}$	$2.6\ 10^{4}$	$2.0\ 10^{3}$
Direct allocation (simplex)	453.	425.	392.
Mixed l_1 optimization	$1.1\ 10^{30}$	98.	324.

Table 8: Identifiability results. C-17 aircraft model with 8 actuators. Infeasible set.

6.5 Identifiability Considerations

Table 8 reflects a different issue from those addressed earlier. It considers the impact that a control allocation algorithm has on the information available to identify the dynamics of an aircraft in real-time (*e.g.*, for reconfigurable control). The analysis is superficial, but the results are interesting enough to report. It has been remarked that the actuator commands produced by most control laws do not excite the system dynamics sufficiently for the estimation of the stability and control derivatives. For example, any control law that applies identical commands to two control surfaces makes it impossible to determine the control derivatives of the separate surfaces. In general, the signals applied to the control surfaces must be linearly independent functions of time in order for identification to be possible. As a result, a technique such as ganging or pseudo-inverse does not provide adequate information for identification. In such cases, one says that the system is not *sufficiently excited*. Given the discrete-time history of a control vector *u(k)*, an indicator of the level of excitation provided (for the determination of the control derivatives) is the autocorrelation matrix

$$R_u = \sum_{k=1}^{N} u(k)u(k)^T \tag{1.46}$$

If any components of the control vector are linearly dependent, the matrix R_u will be singular. In general, a good measure of excitation is then the condition number of the matrix, which we denote CN(R_u). A small condition number is indicative of a large degree of linear independence of the signals, and therefore, a better conditioned identification problem.

Because the vectors a_d used for evaluation in the examples studied before were randomly generated, their time histories constituted linear independent signals. Different control allocation algorithms, however, obtained different control vectors based on these identical requirements. Table 8 shows the condition numbers for the different algorithms, and the same sets of 1000 random vectors. An additional "small" set of accelerations in the range [±8, ±4, ±1] deg/s^2 was added to the earlier feasible and infeasible sets. The C-17 model with 8 actuators was used.

As pointed out earlier, least-squares methods can be expected to give poor results, because they approximate the fixed ganging matrix of the pseudo-inverse when limits are not encountered. The l_1 optimization method can also be expected to give poor results, because it tends to use only the most effective surfaces. The control derivatives of any unused surface cannot be determined. Recent reconfigurable control experiments using these methods [6], [9], [14] have indeed showed the need for separate excitation mechanisms, which typically used random noise injection. Table 8 confirms these observations, but adds some interesting information. Mixed optimization performs poorly for small moment vectors, presumably because some less effective surfaces are not used at all. However, as soon as the vectors are large enough, condition numbers become very small. In that case, all surfaces are used and the special properties of mixed l_1 optimization actually become an advantage.

Interestingly, direct allocation gives excellent results in all three cases. Indeed, the transformation from the required moments to the surface deviations is highly nonlinear, even from small moments (the method does *not* reduce to a generalized inverse around the origin). Also, the fact that the control vectors are scaled from the vectors needed to achieve the maximum moment in the desired direction may be the reason why the condition numbers are so uniform across the range of magnitude. The drawback, of course, is that surface deflections are much larger than needed for feasible moments.

7. Conclusions

Constrained optimization methods can realistically be considered for real-time control allocation in flight control systems. Both direct allocation and mixed error/control optimization problems may be solved with algorithms from linear programming. Computation times are much less than a millisecond, and within an order of magnitude of simpler methods. Gains in performance are small (10% to 20%), but significant. The reassurance that any feasible requirement will be met is valuable, especially in light of the fact that large errors are occasionally observed with simpler methods.

On the downside, linear programming code such as the simplex algorithm is relatively complex. Also, although the ratio of worst-case to average time requirement is small, the variable number of iterations is a disadvantage in flight control applications. Cycling is an issue that may be avoided with anticycling procedures, but remains an issue of concern in the presence of numerical errors. Of course, other methods have their drawbacks too. Large errors are found occasionally with the redistributed pseudo-inverse, large control signals are obtained with the single-pass quadratic programming and with direct allocation, and precision and/or number of iterations vary considerably with the fixed-point method. Perhaps the best results will be obtained in practice by combining a simple method and an optimization-based method, and choosing the best solution of the two. Concerns about reliability could certainly be alleviated this way.

Other optimization algorithms may be considered instead of the simplex algorithm. Interior-point methods are promising, but perhaps for reasons other than those that have made them popular in different applications. The uniform convergence properties of the algorithms towards the optimum solution make them well suited to an environment that requires a fixed bound on the number of iterations. The ability to evaluate the distance from a current estimate to the optimum solution (in primal-dual methods) is also valuable. However, although the algorithms require less computations than the simplex algorithm for very large problems, the picture may actually be reversed for the "small" problems of control allocation. This is a subject of current and future research.

As opposed to simpler methods of control allocation, constrained optimization methods open doors to the more complicated problems of the future. Even simple linear programming algorithms enable the incorporation of constraints between the actuators (*e.g.*, limited differential deflection of tail surfaces, bound on the sum of some reaction jets in spacecraft). More advanced techniques may also extend the range of application to problems where the effectiveness of the actuators is highly nonlinear in the control deflections.

8. References

[1] M. Bodson & J. Groszkiewicz, "Multivariable Adaptive Algorithms for Reconfigurable Flight Control," *IEEE Trans. on Control Systems Technology*, vol. 5, no. 2, pp. 217-229, 1997.

[2] M. Bodson & W. Pohlchuck, "Command Limiting in Reconfigurable Flight Control," *AIAA J. of Guidance, Control, and Dynamics*, vol. 21, no. 4, pp. 639-646, 1998.

[3] J. Brinker & K. Wise, "Reconfigurable Systems for Tailless Fighter Aircrafts -- Restore," Report AFRL-VA-WP-TR-1999-3067, Air Force Research Laboratory, Wright-Patterson AFB OH 45433-7542, 1999.

[4] J. Buffington, "Tailless Aircraft Control Allocation," *Proc. of the AIAA Guidance, Navigation, and Control Conference*, New Orleans, LA, pp. 737-747, 1997.

[5] J. Buffington, "Modular Control Law Design for the Innovative Control Effectors (ICE) Tailless Fighter Aircraft Configuration 101-3," Report AFRL-VA-WP-TR-1999-3057, Air Force Research Laboratory, Wright-Patterson AFB OH 45433-7542, 1999.

[6] J. Buffington, P. Chandler, & M. Pachter, "Integration of On-line System Identification and Optimization-based Control Allocation," *Proc. of the AIAA Guidance, Control, and Dynamics Conference*, Boston, MA, pp. 1746-1756, 1998.

[7] J. Burken, P. Lu & Z. Wu, "Reconfigurable Flight Control Designs with Application to the X-33 Vehicle," *Proc. of the AIAA Guidance, Control, and Dynamics Conference*, Portland, OR, pp. 951-965, 1999.

[8] D. Cameron & N. Princen, "Control Allocation Challenges and Requirements for the Blended Wing Body," *Proc. of the AIAA Guidance, Control, and Dynamics Conference*, Denver, CO, 2000.

[9] D. Doman, A. Ngo, D. Leggett, M. Saliers, & M. Pachter, "Development of a Hybrid Direct-Indirect Adaptive Control System for the X-33," *Proc. of the AIAA Guidance, Control, and Dynamics Conference*, Denver, CO, 2000.

[10] W. Durham, "Constrained Control Allocation," *AIAA J. of Guidance, Control, and Dynamics*, vol. 16, no. 4, pp. 717-725, 1993.

[11] W. Durham, "Attainable Moments for the Constrained Control Allocation Problem," *AIAA J. Guidance, Control, and Dynamics*, vol. 17, no. 6, pp. 1371-1373, 1994.

[12] W. Durham, "Computationally Efficient Control Allocation," *Proc. of the AIAA Guidance, Control, and Dynamics Conference*, Portland, OR, pp. 1310-1320, 1999.

[13] R. Eberhardt & D. Ward, "Indirect Adaptive Flight Control of a Tailless Fighter Aircraft," *Proc. of the AIAA Guidance, Control, and Dynamics Conference*, Portland, OR, pp. 466-476, 1999.

[14] M. Elgersma, D. Enns, S. Shald, & P. Voulgaris, "Parameter Identification for Systems with Redundant Actuators," *Proc. of the AIAA Guidance, Control, and Dynamics Conference*, Boston, MA, pp. 109-117, 1998.

[15] D. Enns, "Control Allocation Approaches," *Proc. of the AIAA Guidance, Control, and Dynamics Conference*, Boston, MA, pp. 98-108, 1998.

[16] Y. Ikeda & M. Hood, "An Application of L1 Optimization to Control Allocation," *Proc. of the AIAA Guidance, Control, and Dynamics Conference*, Denver, CO, 2000.

[17] C. Kim, *Introduction to Linear Programming*, Holt, Rinehart & Winston, New York, 1971.

[18] D. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA 1984.

[19] J. Nocedal & S.J. Wright, *Numerical Optimization*, Springer, New York, 1999.

[20] A. Page & M. Steinberg, "A Closed-Loop Comparison of Control Allocation Methods," *Proc. of the AIAA Guidance, Control, and Dynamics Conference*, Denver, CO, 2000.

[21] J. Petersen & M. Bodson, "Fast Control Allocation Using Spherical Coordinates," *Proc. of the AIAA Guidance, Navigation, and Control Conference*, Portland, OR, pp. 1321-1330, 1999.

[22] J. Petersen & M. Bodson, "Control Allocation for Systems with Coplanar Controls," *Proc. of the AIAA Guidance, Navigation, and Control Conference*, Denver, CO, August 2000.

[23] J. Virnig & D. Bodden, "Multivariable Control Allocation and Control Law Conditioning when Control Effectors Limit," *Proc. of the AIAA Guidance, Navigation, and Control Conference*, Scottsdale, AZ, pp. 572-582, 1994.